**IntelliJ IDEA**

IntelliJ IDEA is a java integrated development environment(IDE) for developing computer

software. It is developed by JetBrains (formerly known as intelliJ).

The first version of IntelliJ IDEA was released in January 2001, and was one of the first

available Java IDEs with advanced code navigation and code refactoring capabilities

integrated.

**System Requirements:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Windows** | **OS X** | **Linux** |
| **OS Version** | Windows 10/8/7 x64 | OS X 10.5 or higher, up to 10.11 (El Capitan) | GNOME or KDE desktop |
| **RAM** | 1 GB minimum; 4 GB or more recommended for Android development, or commercial production. | | |
| **Disk space** | 300 MB hard disk space + at least 1 GB for caches | | |
| **JDK Version** | JDK 1.8 since 2016.1[[11]](https://en.wikipedia.org/wiki/IntelliJ_IDEA#cite_note-11) | | |
| **Screen resolution** | 1024×768 minimum screen resolution | | |

## **IntelliJ IDEA editions:**

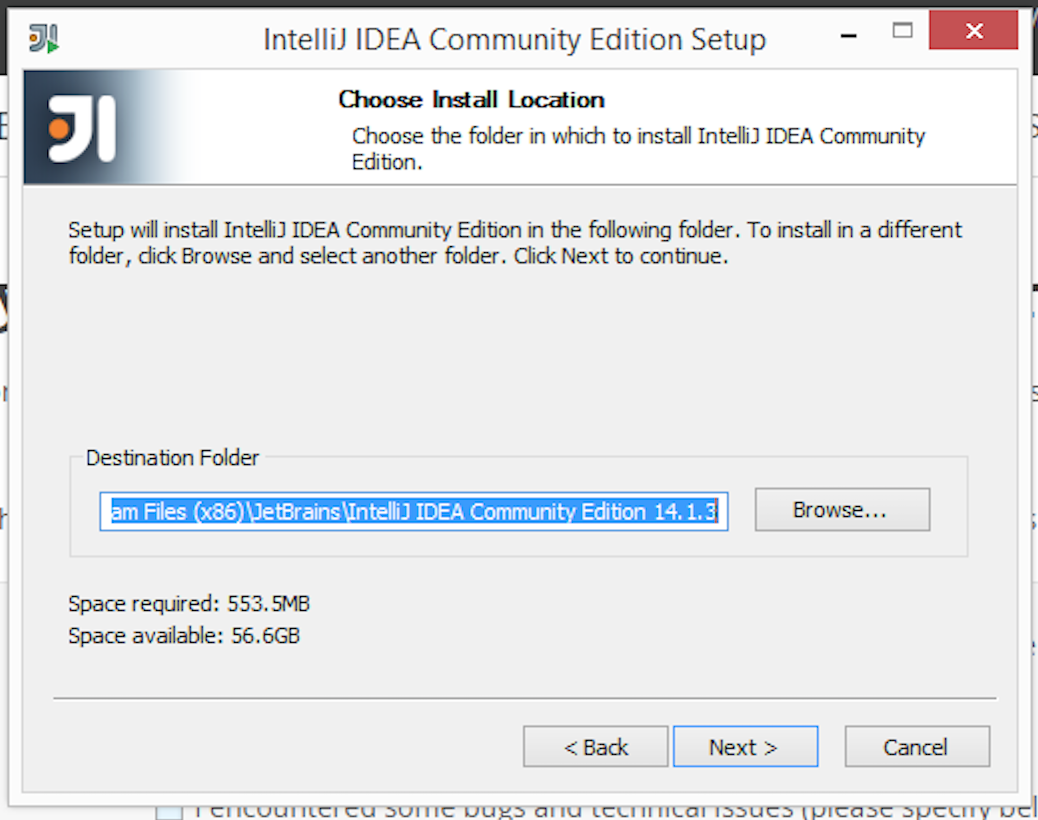
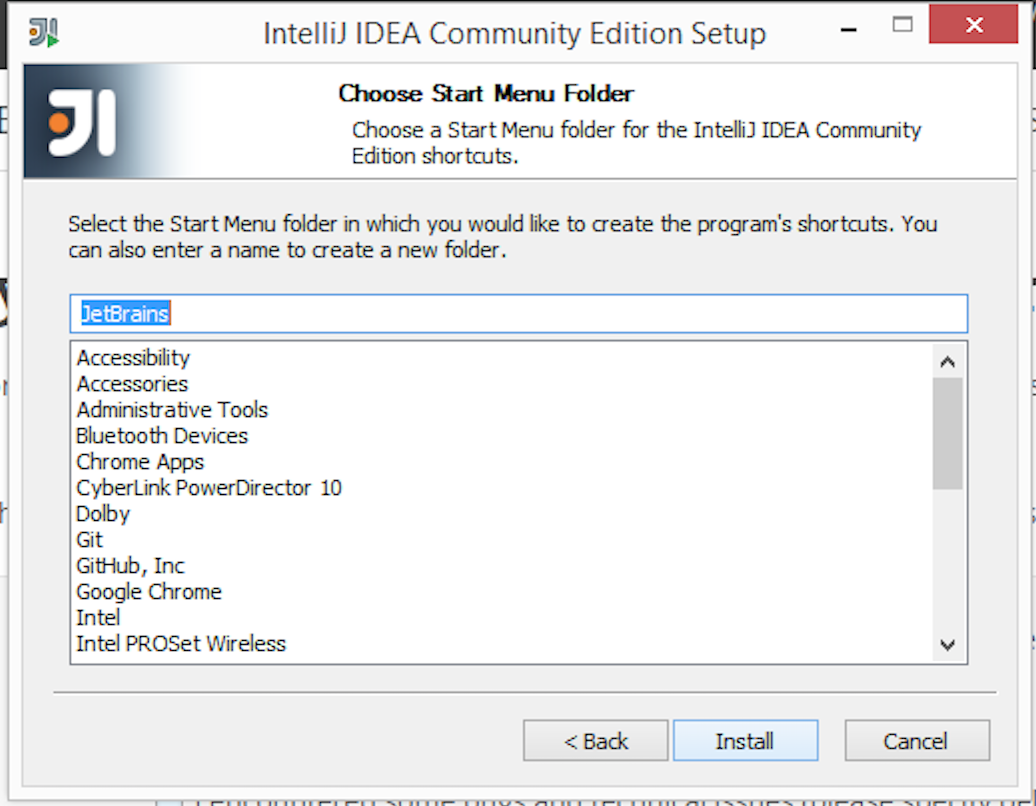
There are two IntelliJ IDEA editions: Community and Ultimate.

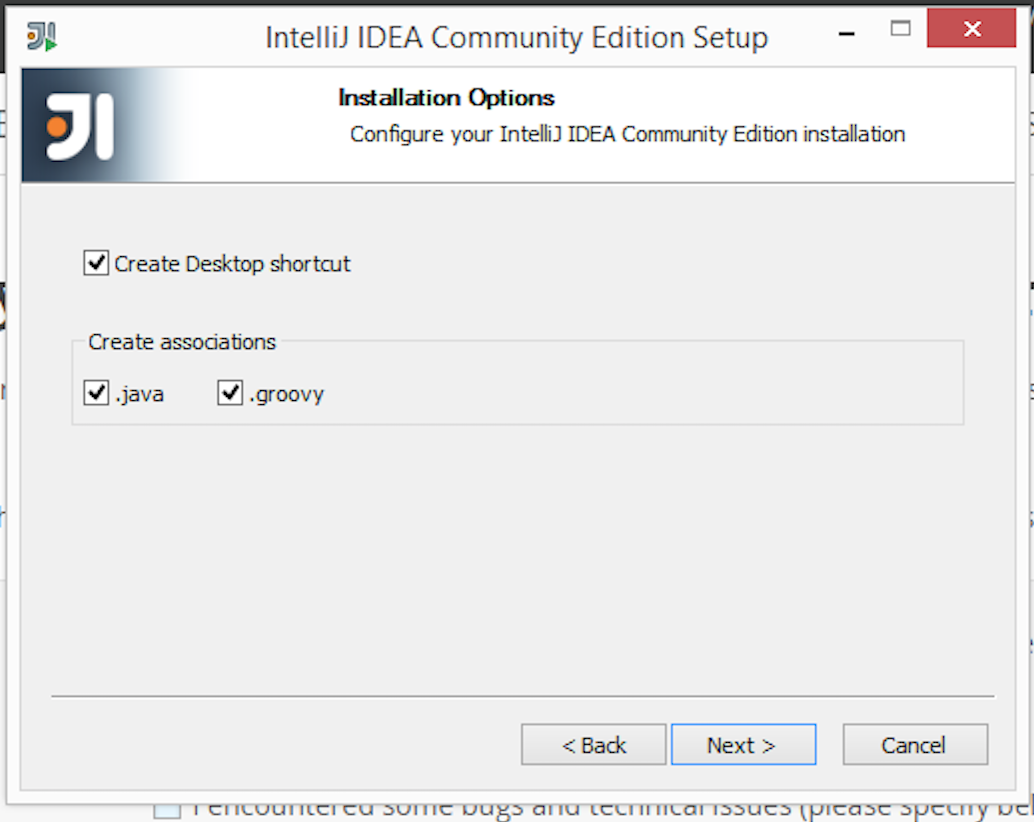
The Community Edition is free and open source but has less features. The Ultimate Edition is commercial but provides an outstanding set of tools and features. (You can evaluate IntelliJ IDEA Ultimate for free for 30 days.)

**Installation Steps:**

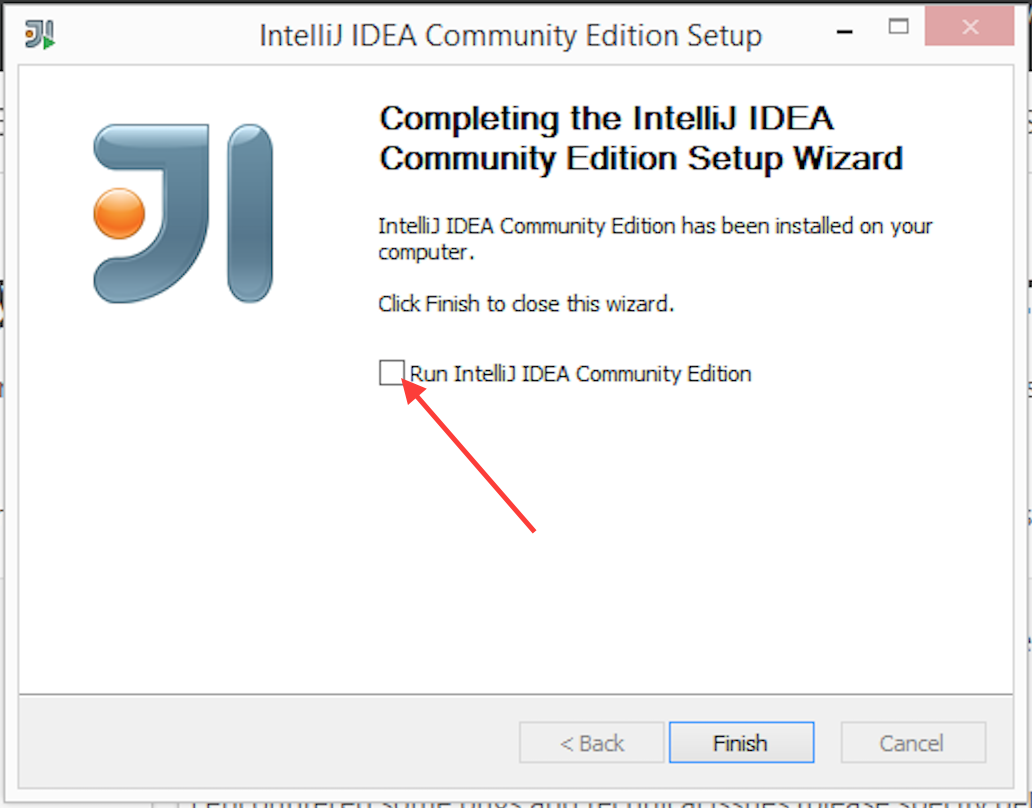
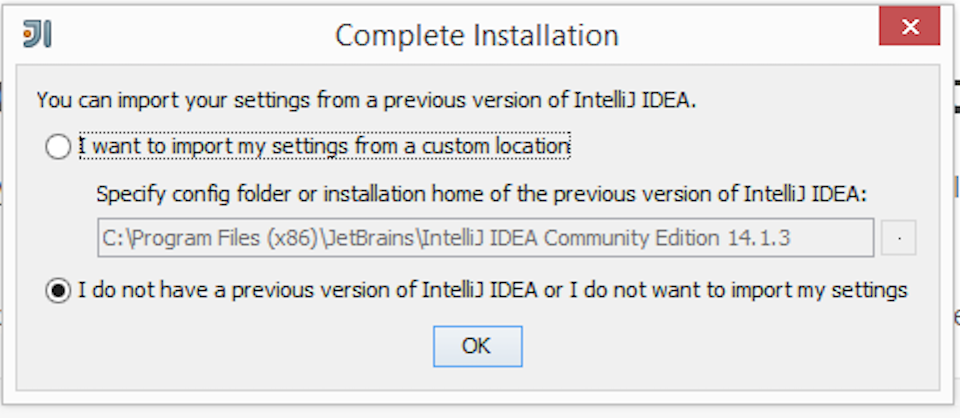
1. Choose the Windows tab (probably defaulted) and click the “Download Community” link
2. Open the .exe and choose next
   * 

3) Choose default install location and menu

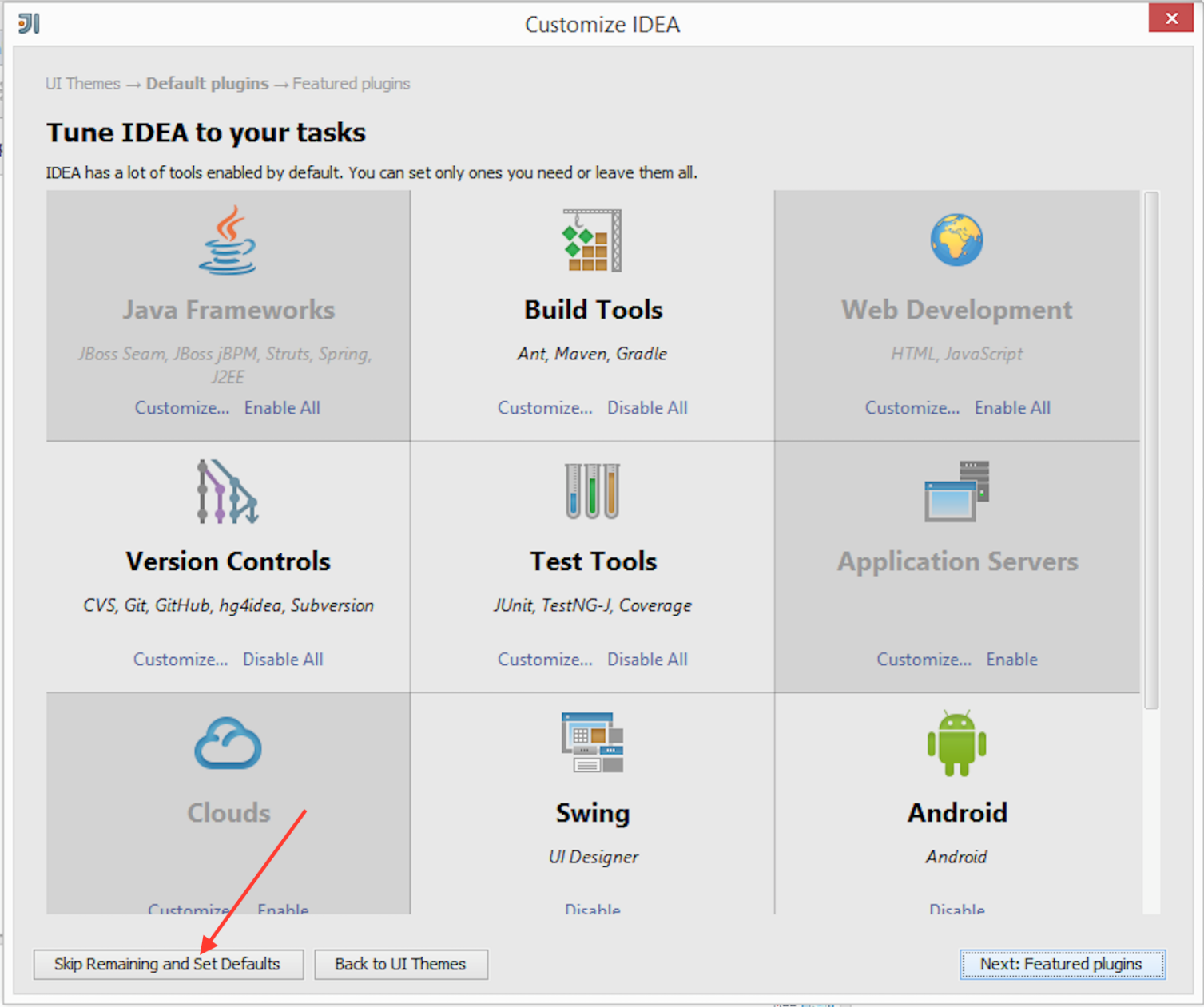
* 
* 

1. Choose to associate .java files (unless you don’t want to, this will make a double click of a .java file open in IDEA)
   * 

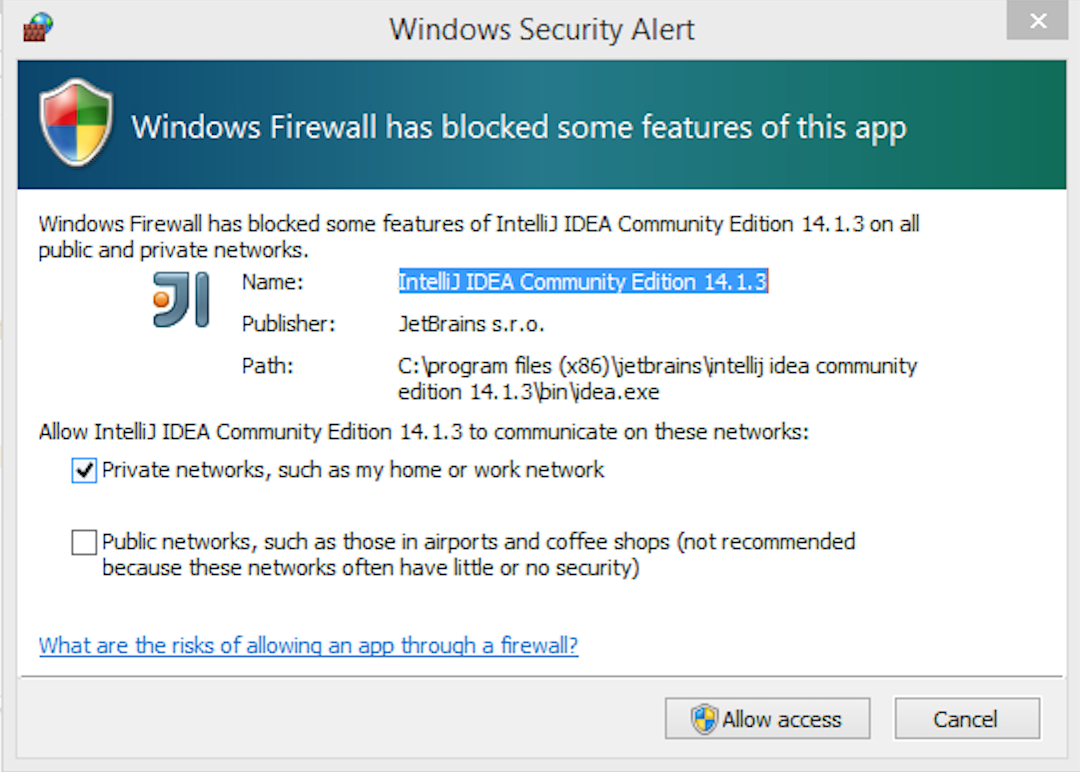
5)Choose the run checkbox, so we can finish installation

* + 
  1. Leave the default chosen for importing (again unless you do have files)
  + 

1. Choose “Skip Remaining and Set Defaults”



1. Finally if you get warned about Firewall access. Choose allow (unless you are feeling paranoid).



**Running Intellij IDEA for the first time:**

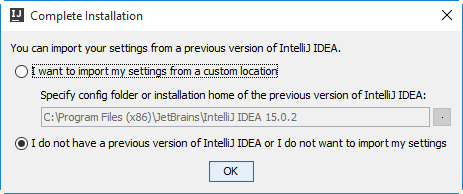
Running IntelliJ IDEA Ultimate for the first time assumes performing the following tasks:

## **1)Importing IntelliJ IDEA settings from a previous version:**

When you start IntelliJ IDEA for the first time, the **Complete Installation** dialog opens,

and you have an option of importing IntelliJ IDEA usage preferences and license information

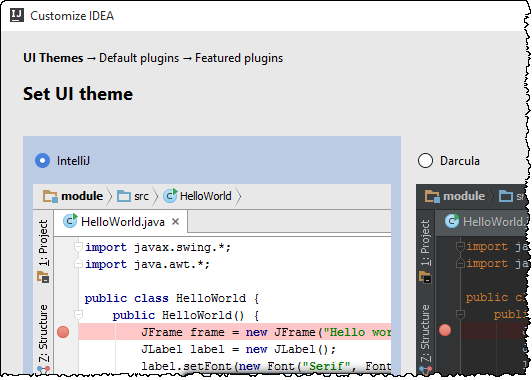
from a previous version.



Select one of the following options and click **OK**.

## **2)Selecting the user interface theme:**

Select the user interface theme to be used and click **Next**.



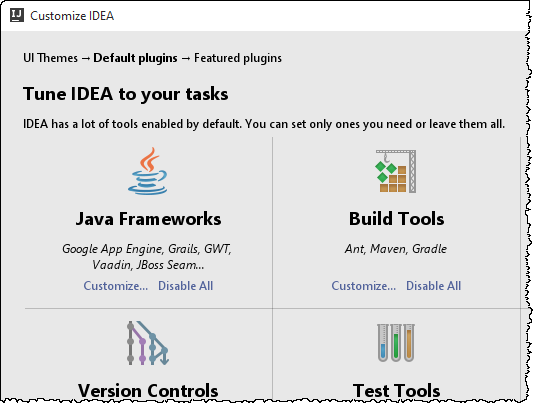
## **3)Disabling unnecessary plugins:**

On the **Default plugins** page, you can disable the plugins that you are not going to use.

(The more plugins are enabled, the more features you have available. On the other hand,

disabling unnecessary plugins may increase the IDE performance, especially on "less

powerful" computers.)



The squares on the page correspond to groups of plugins and individual plugins. All these

plugins are bundled with the IDE and enabled by default.

Use:

* **Customize** to disable one or more plugins in a group. (Use the check boxes on the page

that opens.)

* **Disable All** or **Disable** to disable all plugins in a group or an individual plugin.

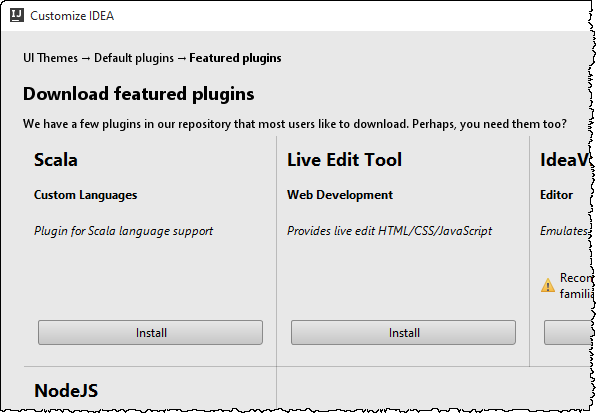
Note that you can come back to disabling and enabling the plugins at a later time.

Click **Next: Featured plugins**.

## **4)Downloading and installing additional plugins:**

The **Featured plugins** page shows the plugins that you might want to download and install.

(All these plugins are not included in the IntelliJ IDEA distribution.)



Click Install to download and install anyone of the plugins.

Click Start using IntelliJ IDEA to complete the wizard. As a result, the Welcome screen

options

**Starting IntelliJ IDEA**

**Windows:**

If during the installation you selected to create a shortcut on your desktop, double-click that

shortcut.

Otherwise, go to the <IntelliJ IDEA>\bin folder (e.g. C:\Program Files

(x86)\JetBrains\IntelliJ IDEA 2017.1\bin) and run idea.exe, idea64.exe, or idea.bat.

Depending on your operating system:

* **macOS:**

 Open the .dmg package, and drag IntelliJ IDEA to the Applications folder.

* **Linux:**

Unpack the .tar.gz archive into any directory within your home directory.

**Shortcuts for Intellij:**

| **Action** | **Shortcut** |
| --- | --- |
| [Find action by name](https://www.jetbrains.com/help/idea/2017.1/navigating-to-action.html) | Ctrl+Shift+A |
| Show the list of available [intention actions](https://www.jetbrains.com/help/idea/2017.1/intention-actions.html) . | Alt+Enter |
| Switch between views ([Project](https://www.jetbrains.com/help/idea/2017.1/project-tool-window.html), [Structure](https://www.jetbrains.com/help/idea/2017.1/structure-tool-window-file-structure-popup.html), etc.). | Alt+F1 |
| [Switch](https://www.jetbrains.com/help/idea/2017.1/navigating-between-files-and-tool-windows.html) between the tool windows and files opened in the editor. | Ctrl+Tab |
| Show the [Navigation bar](https://www.jetbrains.com/help/idea/2017.1/navigation-bar.html). | Alt+Home |
| [Insert a live template](https://www.jetbrains.com/help/idea/2017.1/generating-code.html). | Ctrl+J |
| [Surround with a live template](https://www.jetbrains.com/help/idea/2017.1/creating-code-constructs-using-surround-templates.html). | Ctrl+Alt+J |
| [Edit an item from the Project or another tree view](https://www.jetbrains.com/help/idea/2017.1/opening-and-reopening-files-in-the-editor.html). | F4 |
| [Comment or uncomment a line or fragment of code](https://www.jetbrains.com/help/idea/2017.1/commenting-and-uncommenting-blocks-of-code.html) with the line or block comment. | Ctrl+Slash Ctrl+Shift+Slash |
| [Find class or file by name](https://www.jetbrains.com/help/idea/2017.1/navigating-to-class-file-or-symbol-by-name.html). | Ctrl+N Ctrl+Shift+N |
| [Duplicate the current line or selection](https://www.jetbrains.com/help/idea/2017.1/adding-deleting-and-moving-code-elements.html" \l "duplicate). | Ctrl+D |
| [Incremental expression selection](https://www.jetbrains.com/help/idea/2017.1/selecting-text-in-the-editor.html). | Ctrl+W and Ctrl+Shift+W |
| [Find/replace text string in the current file](https://www.jetbrains.com/help/idea/2017.1/finding-and-replacing-text-in-file.html). | Ctrl+F Ctrl+R |
| [Find/replace text in the project or in the specified directory](https://www.jetbrains.com/help/idea/2017.1/finding-and-replacing-text-in-project.html) | Ctrl+Shift+F Ctrl+Shift+R |
| [Search everywhere.](https://www.jetbrains.com/help/idea/2017.1/searching-everywhere.html) | Double-press Shift |
| [Quick view the usages of the selected symbol](https://www.jetbrains.com/help/idea/2017.1/highlighting-usages.html). | Ctrl+Shift+F7 |
| [Expand or collapse a code fragment in the editor](https://www.jetbrains.com/help/idea/2017.1/code-folding.html). | Ctrl+NumPad Plus Ctrl+NumPad - |
| [Invoke code completion](https://www.jetbrains.com/help/idea/2017.1/auto-completing-code.html" \l "basic_completion). | Ctrl+Space |
| [Smart statement completion](https://www.jetbrains.com/help/idea/2017.1/auto-completing-code.html" \l "statements_completion). | Ctrl+Shift+Enter |
| [Smart completion](https://www.jetbrains.com/help/idea/2017.1/auto-completing-code.html" \l "smart_completion) | Ctrl+Shift+Space |
| Show the list of available [refactorings](https://www.jetbrains.com/help/idea/2017.1/refactoring-source-code.html) (Refactor This). | Ctrl+Shift+Alt+T |

**Migration from eclipse to intellij IDEA**

## **Overview:**

Switching from **Eclipse** to **IntelliJ IDEA**, especially if you've been using **Eclipse** for a long

time, requires understanding some fundamental differences between the two IDEs,

including their [user interfaces](https://www.jetbrains.com/help/idea/2017.1/eclipse.html" \l "user_interface), [compilation methods](https://www.jetbrains.com/help/idea/2017.1/eclipse.html" \l "RunningReloadingChanges), [shortcuts](https://www.jetbrains.com/help/idea/2017.1/eclipse.html" \l "Shortcuts), project configuration and

other aspects.

## **User Interface:**

### No workspace:

The first thing you'll notice when launching **IntelliJ IDEA** is that it has

no **workspace** concept. This means that you can work with only one project at a time. While

in **Eclipse** you normally have a set of projects that may depend on each other, in **IntelliJ**

**IDEA** you have a single project that consists of a set of modules.

If you have several unrelated projects, you can open them in separate windows.

If you still want to have several unrelated projects opened in one window, as a workaround

you can configure them all in **IntelliJ IDEA** as modules.

**IntelliJ IDEA vs Eclipse terminology:**

The table below compares the terms in **Eclipse** and **IntelliJ IDEA**:

| **Eclipse** | **IntelliJ IDEA** |
| --- | --- |
| Workspace | Project |
| Project | Module |
| Facet | Facet |
| Library | Library |
| JRE | SDK |
| Classpath variable | Path variable |

**No perspectives:**

The second big surprise when you switch to **IntelliJ IDEA** is that it has no **perspectives**.

It means that you don't need to switch between different workspace layouts manually to

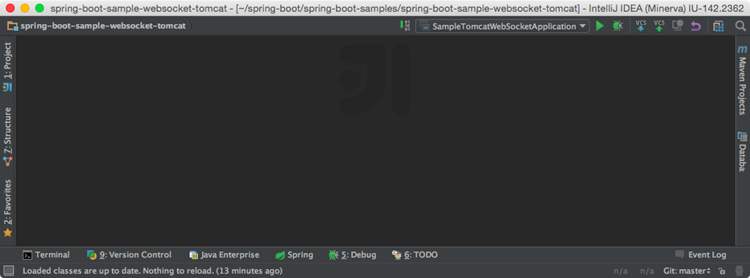
perform different tasks. The IDE follows your context and brings up the relevant tools

automatically.

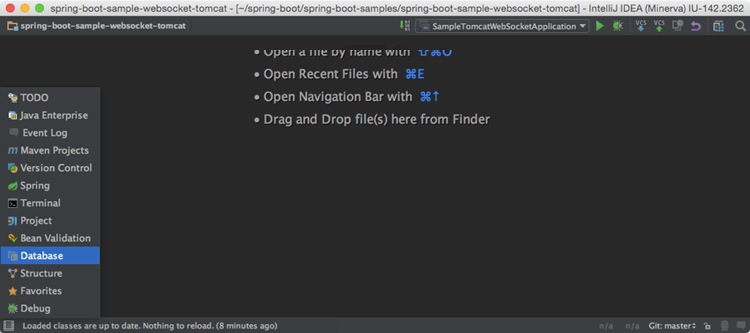
### Tool windows

Just like in **Eclipse**, in **IntelliJ IDEA** you also have tool windows. To open a tool window,

simply click it in the tool window bar:



If the tool window bar is hidden, you can open any tool window by hovering over the corresponding icon in the bottom left corner:



If you want to make the tool window bar visible for a moment, you can press Alt (Cmdfor

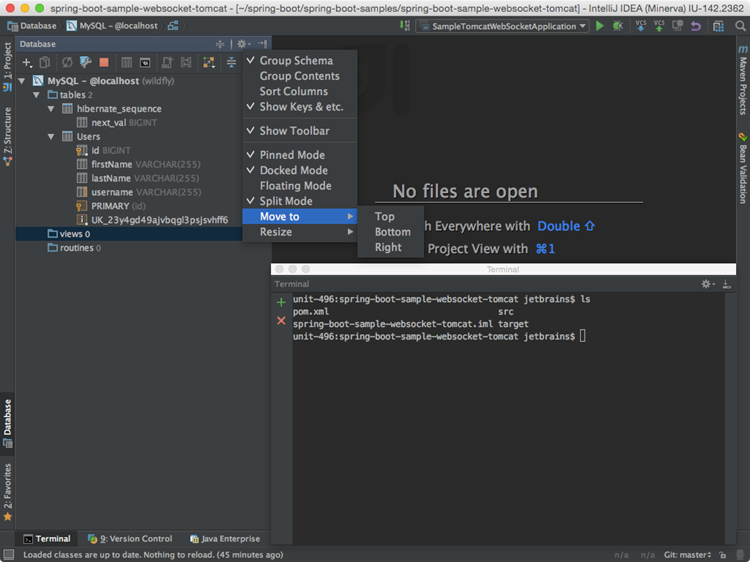
macOS) twice and hold it.

If you don't want to use the mouse, you can always switch to any toolbar by pressing the

shortcut assigned to it. The most important shortcuts to remember are:

* **Project**: Alt+1
* **Version Control**: Alt+9
* **Terminal**: Alt+F12

Another thing about tool windows is that you can drag, pin, unpin, attach and detach them:



To help store/restore the tool windows layout, there are two useful commands:

* **Window | Store Current Layout as Default**
* **Window | Restore Default Layout** (also available via Ctrl+F12)

### Multiple windows

Windows management in **IntelliJ IDEA** is slightly different from **Eclipse**. You can't open

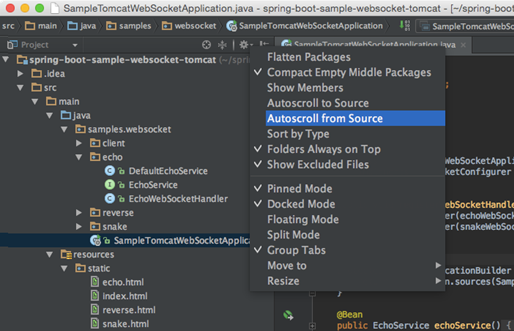
several windows with one project, but you can detach any number of editor tabs into

separate windows.

### Auto-scrolling to/from sources

By default, **IntelliJ IDEA** doesn't change the selection in the [Project tool window](https://www.jetbrains.com/help/idea/2017.1/project-tool-window.html) when you

switch between editor tabs. However, you can enable it in the tool window settings:



### Enabling line numbers

Line numbers are not shown in the editor by default. To enable them, go

To **Settings/Preferences | Editor | General | Appearance | Show line numbers**. There

you will also find other useful settings.

## **General workflows**

### No 'save' button

Time for some really shocking news: **IntelliJ IDEA** has no **Save** button. Since in **IntelliJ**

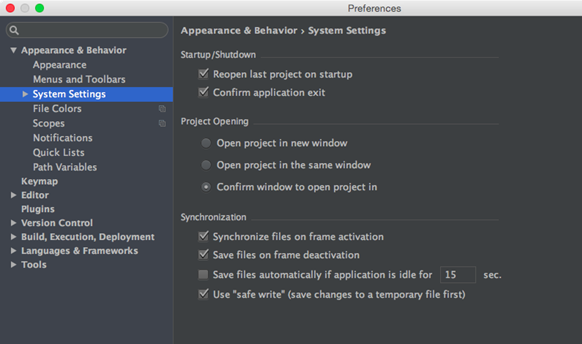
**IDEA** you can undo refactorings and revert changes from [Local History](https://www.jetbrains.com/help/idea/2017.1/local-history.html), it makes no sense

to ask you to save your changes every time.

Still, it's worth knowing that physical saving to disk is triggered by certain events, including

compilation, closing a file, switching focus out of the IDE, etc. You can change this behavior

via **Settings | Appearance & Behavior | System Settings**:



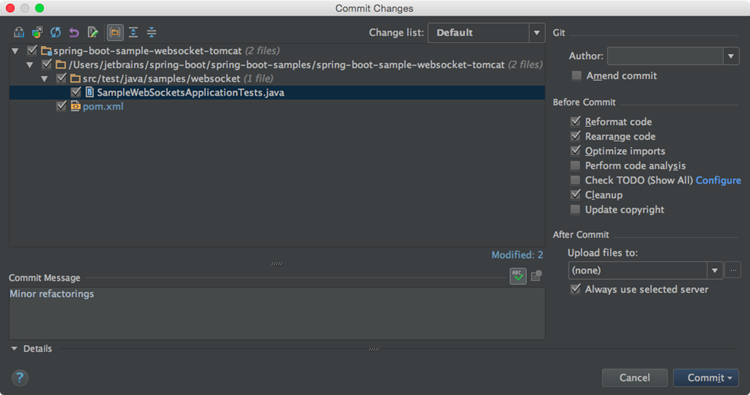
### No save actions

One of the features you may miss in **IntelliJ IDEA** as an Eclipse user is **save actions**, i.e.

the actions triggered automatically on save, such as reformatting code, organizing imports,

adding missing annotations and the **final** modifier, etc. Instead, **IntelliJ IDEA**offers you to

run the corresponding actions automatically on commit:



Or manually:

* **Code | Reformat Code** (Ctrl+Alt+L)
* **Code | Optimize Imports** (Ctrl+Alt+O)
* **Analyze | Code Cleanup**

If, for some reason, you can't live without an **Eclipse** save action, you can install [plugin that imitates Eclipse save actions](https://plugins.jetbrains.com/plugin/7642).

### Compilation:

The way **IntelliJ IDEA** compiles projects is different from **Eclipse** in a number of ways.

#### **Auto-compilation:**

By default, **IntelliJ IDEA** doesn't automatically compile projects on saving because

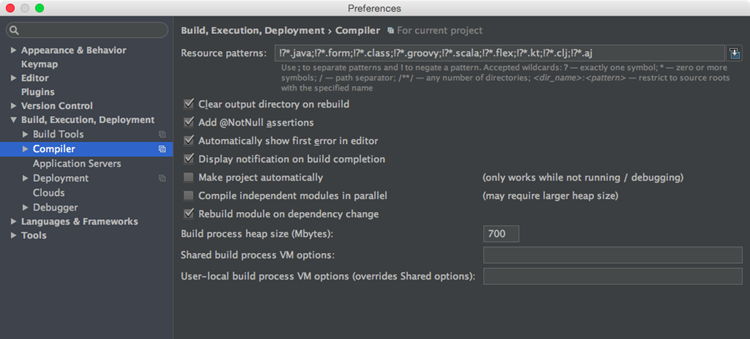
normally we don't invoke the **save** action explicitly in **IntelliJ IDEA**.

If you want to mimic the **Eclipse** behavior, you can invoke the **Make Project** action

(Ctrl+F9) - it will save the changed files and compile them. For your convenience, you can even reassign the Ctrl+S shortcut to the **Make Project** action.

To enable automatic compilation, navigate to **Settings/Preferences | Build, Execution,**

**Deployment | Compiler** and select the **Make project automatically** option:



Note that automatic compilation in **IntelliJ IDEA** differs from that in **Eclipse**. In **Eclipse**it's

not fully automatic, as it is triggered by the **save** action invoked by the user explicitly,

whereas in **IntelliJ IDEA** it is invoked implicitly when you type in the editor.

This is why, even if the **Make project automatically** option is enabled, **IntelliJ**

**IDEA**doesn't perform automatic compilation if at least one application is running: it will

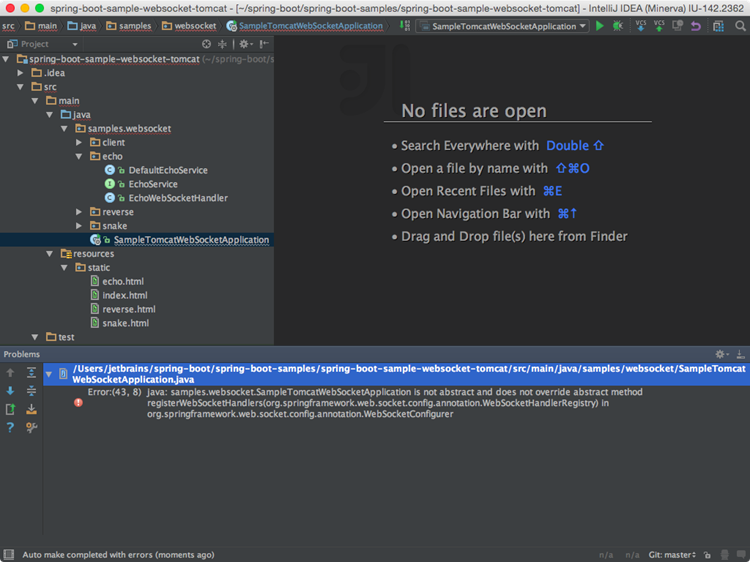
reload classes in the application implicitly. In this case you can call **Build | Make**

**Project** (Ctrl+F9).

#### **Problems tool window**

The [Problems tool window](https://www.jetbrains.com/help/idea/2017.1/problems-tool-window.html) appears if the **Make project automatically** option is enabled in

the [Compiler settings](https://www.jetbrains.com/help/idea/2017.1/compiler.html). It shows a list of problems that were detected on project compilation:



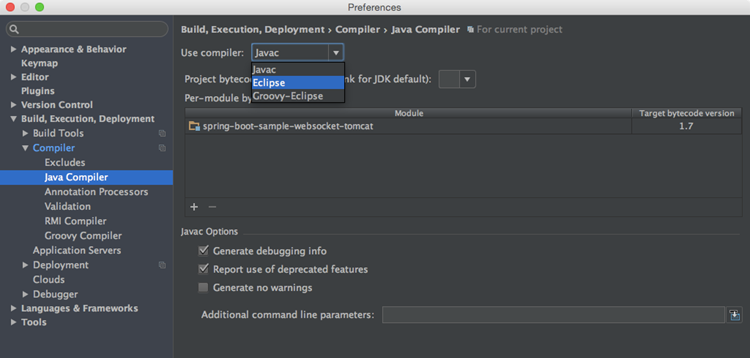
#### **Eclipse compiler**

While **Eclipse** uses its own compiler, **IntelliJ IDEA** uses the **javac** compiler bundled with

the project JDK. If you must use the **Eclipse** compiler, navigate to **Settings/Preferences**

**| Build, Execution, Deployment | Compiler | Java Compiler** and select it as shown

below:



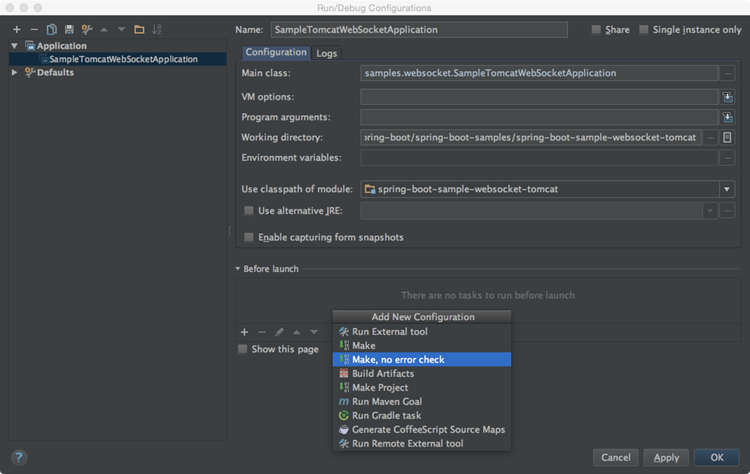
The biggest difference between the **Eclipse** and **javac** compilers is that the **Eclipse**

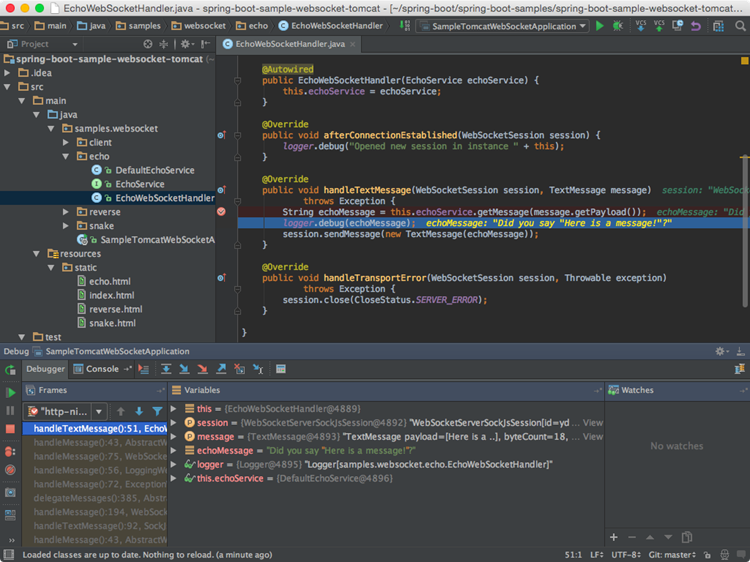
compiler

is more tolerant to errors, and sometimes lets you run code that doesn't compile.

In situations when you need to run code with compilation errors in **IntelliJ IDEA**, replace

the **Make** option in your [run configuration](https://www.jetbrains.com/help/idea/2017.1/run-debug-configurations.html) with **Make, no error check**:





**IntelliJ vs Eclipse**

* **Plugins:** Eclipse marketplace offers 1,276 plugins, and the Intellij Plugin Repository

offers 727 plugins. This difference is not to be taken lightly, since plugins for new

technologies will usually be developed mainly for Eclipse (e.g. Android, Drools, Activiti,

etc). Moreover, Eclipse is easier to extend. When working on a specific technology most

chances are that if a plugin exists, it will be an Eclipse plugin.

* **Multiple projects:** This is an Eclipse winner for sure. It has the ability to open multiple

projects in the same window, giving the coder control over dependencies and relations.

Intellij has an option to open one project with multiple modules, but we found it to be

cumbersome, and in times a little buggy. If you are going to use a lot of projects

together and hate to switch windows, Eclipse is your choice.

* **Multiple languages:** We have stated that we will only examine the Intellij Community

Edition that supports Java, Groovy and Scala. However, if you plan to create a Python

server, combined with Ajax & Html, joint with a java web server, or any other exotic

language combinations, than Eclipse is your choice.

* **Code completion & inspection:** While Eclipse has the ability to add plugins such

as [checkstyle](http://eclipse-cs.sourceforge.net/), this one definitely goes for Intellij. The default code completion and

assistance in Intellij is faster and better. If you are a rookie developer, Intellij can

improve your code.

* **Usability:**Intellij user experience is much easier to grasp. The learning curve in Intellij

is by far faster. It seems using Intellij makes developing easier and more natural.

Dropdowns, code completion, quick view, project wizards, etc, are all possible both in

Eclipse and Intellij, but the experience in Intellij is much more satisfying.

* **Performance:** The more plugins are installed on the IDE, the more heavy it is for your

computer. However, saying that, Eclipse handles very large projects faster. Moreover,

both of the IDE’s seems to be RAM junkies. Projects usually open faster in Eclipse, as

Intellij indexes the entire project on startup, but while working on an existing project,

Intellij works smoother. For example we have a huge SOAP project, which is impossible

to work on with Intellij, so some of us even learn Eclipse just for that.

* **Repository integration:** Both of the IDE’s have SVN\GIT\etc plugins. No doubt Intellij’s

plugin is more reliable, has better GUI and easier to use.

* **GUI builder:** We found that the built in Intellij GUI builder is more comfortable, and as
* mentioned above, usability wise its easier to learn, and more enjoyable to develop.

**For a conclusion**, a programmer should be able to find the right tool given a specific task.

This means that one should be acquainted with both of the IDE’s, in order to face the challenge

with the right tool.