Scala & Framework installation guide

We recommend you use one of the following ways to develop Scala code.

- Using the IntelliJ IDE
- Using the command line to build and a text editor (vscode, sublime,gedit, vim, emacs etc)

The IntelliJ IDE is nice, but can be pretty slow. We recommend that you use the IntelliJ IDE if your computer is fast enough, and otherwise use the command line + text editor.

To use the IntelliJ IDE you need two things:

- A java installation, namely Java Development Kit (JDK) 8
- The IntelliJ IDE

If you develop using the command line + text editor you only need the Java installation.

After installing, you can install the framework for exercise 1
Frame work exercise 1

Video of how to set up exercise 1

Notes

IMPORTANT: A library we are using (Processing) requires JDK8 (nothing newer!) so if you have another version, do not attempt to skip the Java installation steps. If problems arise, please make attempts to remove other versions (or try to fight the conflicts). If all else fails, use an Ubuntu 18.04 VM (using VirtualBox or similar software).

Installing IntelliJ with Scala

GNU+Linux (Ubuntu)

Make sure your version of Ubuntu is greater or equal to 16.04.

Install Java

1. Check your version of Java

```
$ java -version
openjdk version "1.8.0_xxx"
...
```

2. Check your version of JDK

```
$ javac -version
javac 1.8.0_xxx
```

3. If these don't match then run this command:

```
sudo apt install openjdk-8-jdk openjdk-8-jre
sudo update-alternatives --config java
sudo update-alternatives --config java
```

(pick java 8/1.8 when prompted)

Install IntelliJ IDEA

1. Install Snap: `sudo apt install snap`

2. Get the IntelliJ IDEA Community Edition Snap

   ```
   sudo snap install intellij-idea-community --classic
   ```

   - If you encounter the featured plugins picker at the first launch, you can install the Scala plugin right then and there.

Set up Scala in IDEA

1. On the menu bar, go to the hamburger menu, then `File -> Settings -> Plugins -> Marketplace`
2. Find the Scala plugin
3. Install

macOS

We assume you have macOS Sierra or greater. macOS comes with the wrong JRE & JDK so we'll need to get OpenJDK versions of the Java toolchain. To make your life easier you should get Homebrew (if you don't have it already).

Install Homebrew

1. Launch your terminal
2. Run this:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

3. Check if the install was successful by running: `brew info`. It should return some output.

Install Java

1. Check your version of Java

   ```
   $ java -version
   openjdk version "1.8.0_xxx"
   ...
   ```

2. Check your version of JDK

   ```
   javac -version
   javac 1.8.0_xxx
   ```

3. If these do not start with 1.8.0 then run these commands:

   ```
   brew tap adoptopenjdk/openjdk
   brew install --cask adoptopenjdk8
   export JAVA_HOME=`/usr/libexec/java_home -v 1.8`
   ```

4. Check your version of Java now.

IntelliJ IDEA

1. Get the Community Edition **here**
2. Run the installer (basically, click next until done)
   - If you encounter the featured plugins picker at first launch, you can install the Scala plugin right then and there.

OR

```
$ brew install --cask intellij-idea-ce
```

Set up Scala in IDEA

1. On the menu bar, go to `IntelliJ IDEA -> Settings -> Plugins -> Marketplace`
2. Find the Scala plugin
3. Install

Make sure the path to your project does not contain spaces, or things will go awry on macOS

Windows

Install Java

1. In the **command prompt** (cmd or powershell), check your version of JRE:

   ```
   C:\> java -version
        openjdk version "1.8.0_xxx"
        ...
   ```

2. Check your version of JDK

   ```
   C:\> javac -version
        javac 1.8.0_xxx
   ```

3. If this version is < 1.5 You might have to uninstall java from your PC first, this is usually in %JAVA_HOME%. But some earlier versions are installed in C:\Windows\System32.

   If your Java version is not 1.8.0_xxx

   - Get openJDK8 **here**
   - Install it

4. Restart your command prompt
5. Check your version of Java now

IntelliJ IDEA

1. Get the ***Community*** Edition **here**
2. Run the installer (basically, click next until done)
   - If you encounter the featured plugins picker at first launch, you can install the Scala plugin right then and there.

Set up Scala in IDEA

1. On the menu bar, go to `File -> Settings -> Plugins -> Marketplace`
2. Find the Scala plugin
3. Install
4. Test it out by creating a new project in IDEA, select Scala library (project SDK), press download. If error cannot find java program. Set JAVA_HOME variable

Setting JAVA_HOME environment variable

More detailed guide **here**

1. Go to advanced system settings
2. Click the "Environment Variables…" button on the bottom right.
3. Click the New button under System Variables (lower half)
4. In the new window "Edit system variable"
   - Set "variable name" to `JAVA_HOME`
   - Set directory to jdk root folder. This is typically in somewhere like `C:\Program Files\adoptopenjdk\jdk...`

Importing & Using the assignment frameworks

Download links

Download the framework for your exercise:

Warmup exercise:
https://github.com/VU-Programming/OOFP-warmup/archive/refs/heads/master.zip

Selector example: https://gitlab.com/vu-oofp/gamebase/-/archive/select/gamebase-select.zip

Sokoban example:
https://gitlab.com/vu-oofp/gamebase/-/archive/sokoban/gamebase-sokoban.zip

Snake exercise:
https://github.com/VU-Programming/OOFP-snake/archive/refs/heads/master.zip

Tetris exercise :
https://github.com/VU-Programming/OOFP-tetris/archive/refs/heads/master.zip

REPL exercise :
https://github.com/VU-Programming/OOFP-repls/archive/refs/heads/master.zip

Frameworks setup

In `test/` the tests are listed (Read the tests here, the code in the infrastructure does not contain tests)
In `src/` the source code is listed (that you have to modify).

The entry point where you have to start coding is as follows:

Warmup: `src/warmup/Exercises.scala`
Snake: `src/main/snake/logic/GameLogic.scala`
Tetris: `src/main/tetris/logic/TetrisLogic.scala`
REPLs: `src/repls/IntRepl.scala` (for 4.1)

In IntelliJ

Note that you can also do everything listed under command line, i.e. to run specific Gradle tasks

Setup

1. Select new ... -> Project from existing sources
2. Select directory of the framework (i.e. `warmup-master`) Do not select the directory (do not select the directory where warmup-master resides, choose the directory itself)
3. When prompted asking “Import project from external model”, select **Gradle** and press finish (If this is not pre-selected to Gradle, then probably you picked the wrong directory).

You might now see an empty screen. Press the file icon at the top left to see the files.

Running the tests

To run the tests, navigate to a test class in `tests/` and press the play button next to it. You can also run/debug a specific test by pressing the play button next to the test.

If you run the main test class for your exercise (for example `AllTests.scala` for assignment 1)

If running/debugging a single test does not work (instead all tests in the class are executed), do the following:

1. Go to the hamburger menu: `File -> Settings -> Build, Execution, Deployment -> Build Tools -> Gradle`
2. Change *Run tests using* from Gradle to Intellij IDEA
3. Press apply

Zipping your code for submission

1. From the menu bar select Run and then `Run...`
2. From the menu that appears select `Edit configurations`
3. Press the + button at the top left

4. Select `Gradle` from the menu that appears
5. Use the following settings:
    a. Use classpath of gradleproject = <name of your project, probably snakeskeleton>
    b. task = `zip`
6. Press run
7. A zip file will appear in the root directory of your project, hand this in via canvas

Setting up the project SDK for game exercises

In the gaming exercises, you must use JDK 8 (in the warmup and repl exercises it does not matter)

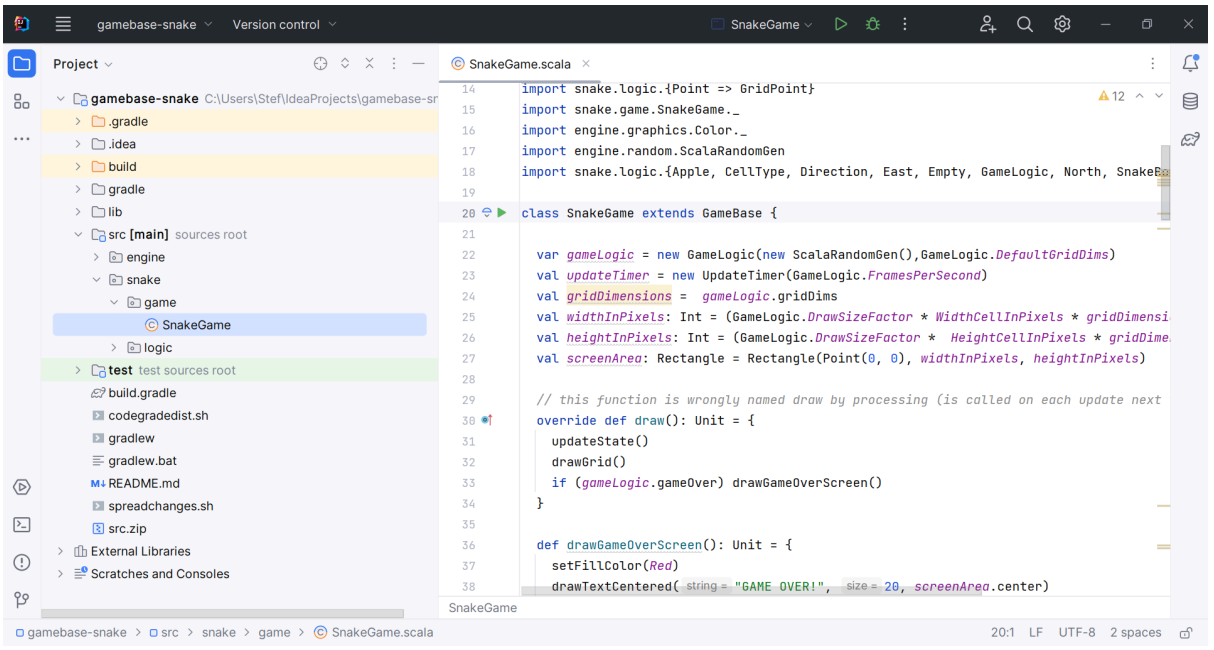
1. Install the OpenJDK 1.8 as described above if you did not already do so.
2. Press setup JDK in the blue warning bar above your code, or go to **File** -> **Project Structure** -> **Project tab in Project Settings**
3. From the dropdown box, select the JDK version 1.8. If this JDK is not present, press the **New..** button next to the dropdown
    - From the emerging dropdown after pressing new, select the **JDK option**. Now navigate the the OpenJDK 1.8 install folder and select it (open it)
        - for **Windows:** C:/ProgramFiles/AdoptOpenJDK/...
        - for **macOS:** /Library/Java/JavaVirtualMachines/...
    - Now the OpenJDK 1.8 is available in the SDK dropdown, select it if this is not automatically done.
4. At the Project Language Level now select level 8 from the dropdown.
5. Hit apply and OK and your JDK and SDK should now be properly set up.

Running the game

For Snake/Tetris/Sokoban Running the Snake game should be fairly easy:

1. Open the main game file
    a. src/`main/snake/game/SnakeGame`
    b. src/`main/snake/game/SnakeGame`
    c. src/`main/tetris/game/TetrisGame`

2. And press the play button left of to **class**



On the command line

Navigate to the directory where you put the framework

You can use the following commands (on windows you need to type gradlew instead of ./gradlew):

`./gradlew test` (builds the code and runs the tests)

(You can also run specific tests as follows:
`gradle test --tests AverageGradeTests.testaverage* ).`

If you want to stop testing after the first failing test use `--fail-fast`

`./gradlew run` (If applicable: run the game. Does not work in warmup/repl exercises)

`./gradlew zip` (zips your code for submission on canvas/codegrade) A zip file will appear in the root directory of your project, hand this in via canvas

To run the snake tests for 2.1 or 2.3 do ./gradlew test2_1 (or ./gradlew test2_3)

Troubleshooting

Game crashes immediately after window appears

If you run the game and the game crashes immediately after the game window appears, then you've run into a very specific bug. (This happened to 1 student)

The reason is that there is some problem with graphics acceleration on some systems. To fix this, turn off acceleration by changing a line of SnakeGame.scala:

from:

```
size(widthInPixels, heightInPixels, PConstants.P2D)
```

to

```
size(widthInPixels, heightInPixels)
```

And then try again