Introduction Exercise

Objective

The goal of this introduction exercise is to prepare you for the course project, which will account for 40% of your final score, by getting you familiarized with the necessary tools for mobile game development.

Technology

You can do the exercises with either android or iOS. You stand free to choose any framework or gaming library for the exercises and project long as it doesn't completely restrict your architecture, such as an intricate game engine would.

For android development, it is suggested you use the libgdx framework or if you want to develop a native application, you can use the SurfaceView class provided by the android sdk, to draw game objects. But feel free to choose a different approach.

A guide to get started with the libgdx framework will be provided.

Delivery

You should zip and deliver all the code and files together with either an apk file (for android) or a runnable jar file (for libgdx).

TASK 1 – Sprites

For this task, you should draw graphics on the screen. You can use the helicopter texture or your own texture for the sprite.

Draw the helicopter on the screen and make it move around on its own. If the sprite is about to leave the screen, it should "bounce" off the edges and head in the opposite direction. Also, make sure the helicopter faces the direction it is traveling.

TASK 2 – Input and text

Reuse the helicopter sprite from task 1.

   a) Draw the sprite on the screen, but instead of having it go around on its own, make it so that you can use a touch function to control the movement. The sprite must not be allowed to exit the screen.

   b) Print the position of the sprite (with screen coordinates). The text should be drawn in the upper-left corner of the screen.

TASK 3 – Animation, timing, and collision detection

In this task, use the helicopter sprite sheet. The sprite sheet contains the animation frames for the helicopter, simulating the spinning of the rotor blades. Each frame is 130 pixels wide and 52 pixels high.
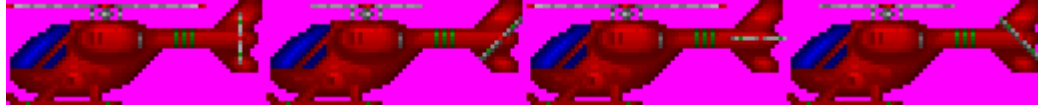
a) Create the same scenario as in task 1, but apply animation to the sprite. Each frame in the animation should display for 100 ms before changing to the next frame. Also, the animation should loop, meaning that you start around again with the first frame after the last one finished.

b) Throw in a couple more sprites (you can use the same sprite, or create new ones yourself). Move the sprites around in random directions and with random speed. If the sprites collide, i.e. their bounding volumes intersect, the sprites should bounce off each other.

TASK 4 – Create Pong

Build on what you have learned in the previous tasks, and implement your own version of the classic game Pong. The game should follow the following basic rules, but feel free to apply your own twists to it:

- There are two paddles, one on each side of the screen. The paddle can be moved vertically and should be kept inside the screen. It is up to you whether both paddles should be controlled by humans (multiplayer), or one of the paddles should be controlled by the computer (single player).

- The ball should start off in the center of the screen and bounce off the paddles, and the upper and lower edges of the screen.

- If the ball slips past a paddle, the paddle on the opposite side is given one point, and the ball is reset to the middle of the screen. The score should be displayed somewhere on the screen. First one to reach 21 wins the game.

Hints and tips:

- Experiment with the speed of the ball and paddles to make the game challenging. It should not be too easy to reach the ball; neither should it be too hard.

- To make things even more challenging, try increasing the speed of the ball after a certain time if no one scores a point, or if the ball hits the edges of the paddles
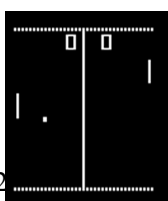
Android Studio Guide

This guide shows you how to instal Android Studio. It is recommended to use Android Studio when writing Android applications. Further down in the document you will find a guide on how to set up LibGDX. It can be beneficial to use LibGDX for solving the exercise, especially if you are a beginner in Android.

Step 1 - Install Java Development Kit (JDK)

Java 9 has recently been launched, but Android does not offer support for Java 9 yet. Therefore it is recommended to download the JDK for Java 8. Use the following link for the download.

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Step 2 - Download Android Studio

Android Studio is the most popular IDEA for creating Android Applications. You are free to use another IDEA such as Eclipse or IntelliJ, but the setup for Android will then become significantly more difficult.

Use the following link to download Android Studio:

https://developer.android.com/studio/index.html

When the installation starts, accept all default values and press "install". After the installation finishes you are free to start Android Studio! Before you can begin programming you will be presented with the Android Studio Setup Wizard. Accept all default values (unless you have some special preferences).

Step 3 - New Android Project

When the Setup Wizard finishes, press: "Start a new Android Studio Project". You will be prompted to a window where you can write the name of your app, company domain and choose a Project Location. You can leave the Company Domain as it is, and remember for later where you saved your new project. In this way you will not lose overview when you create several projects.

Again, accept all values until you get to the "Add an Activity"-window.

Steg 4 - Add an Activity

When you get to the following window (image below) you will be asked to add an activity.
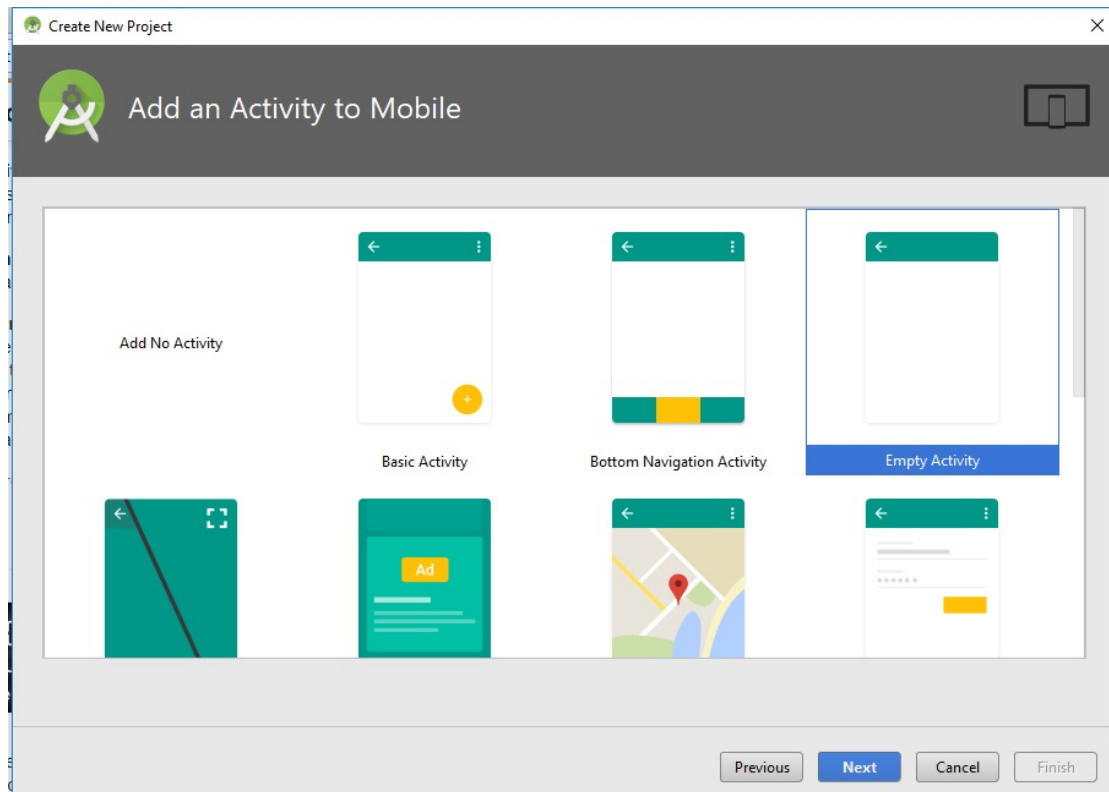
In LibGDX activities are done differently, however it can still be good to know what a classic Android Activity is. You can think of an Activity as a window that your users sees. For example, a classical "log in"-window.

Choose "Basic Activity" this time. This will lead you to the "Configure Activity"-window, here you don't need to change any values.

Step 5 - Fix potential errors

When you have created a project, and picked an activity, you could get some errors. For example, you could get the error: "Gradle (prosjektNavn) project refresh failed"

Press the following link "Install missing platform(s) and sync project". This will hopefully install the missing platform!

After that error is fixed, a new similar error could appear. Again, press the link next to the error to install the missing platform. If you get different (or more) problems, don't be afraid to ask for help!

Step 6 - Get to know Android Studio

In the image below, there are three blue boxes showing the first things you should become familiar with. Again, these things are not present in the same way if you use LibGDX, but are still nice to know for later.
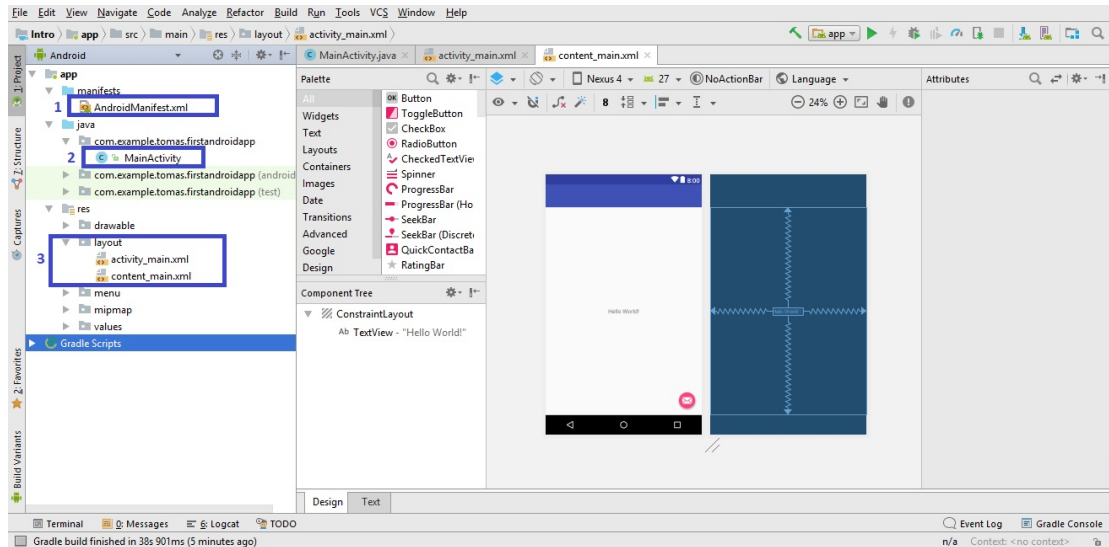
The three things marked on the image are:

1) AndroidManifest.xml - This file contains XML that shows information about your app.
2) MainActivity - Contains the first code to be run when the application starts.
3) Layout - If you double-click on "content_main.xml" in the Layout-folder the design-windows for Android will show up to the right. Here you can add buttons, checkBoxes and etc. to your app. You can toggle between "Design" and "Text" to switch between designing with a user interface, to designing by only using XML-files.

Step 7 - Create a virtual device

When testing how your Android app runs you can either connect your Android phone via USB to your computer, or you could use an Android emulator (virtual device). When you press Shift + F10 to run your app, a window will show up that allows you to choose a virtual device. Create a new virtual device if you don't have one already. You are free to pick any options you wish for your virtual device.
That was it! You are now ready to start writing your app, or continue on to the LibGDX-guide.

LibGDX Guide

LibGDX offers functionality that can make animations and movement easier to implement. Especially if you are a beginner in Android it can be easier to do this exercise in LibGDX. In addition, if you are making a game with collisions, and movement (such as in the first exercise) for the main project, using a library such as LibGDX will significantly simplify the process.

This tutorial-series on YouTUbe teaches how to make FlappyBird in LibGDX.

A lot of the concepts covered in the series are useful for solving exercise 1, and in later game development! You can watch the video to learn how to set up LibGDX, instead of reading this guide. However, the video does not cover step 10. Therefore, if you get problems with running your application, you should look at step 10 in the guide.
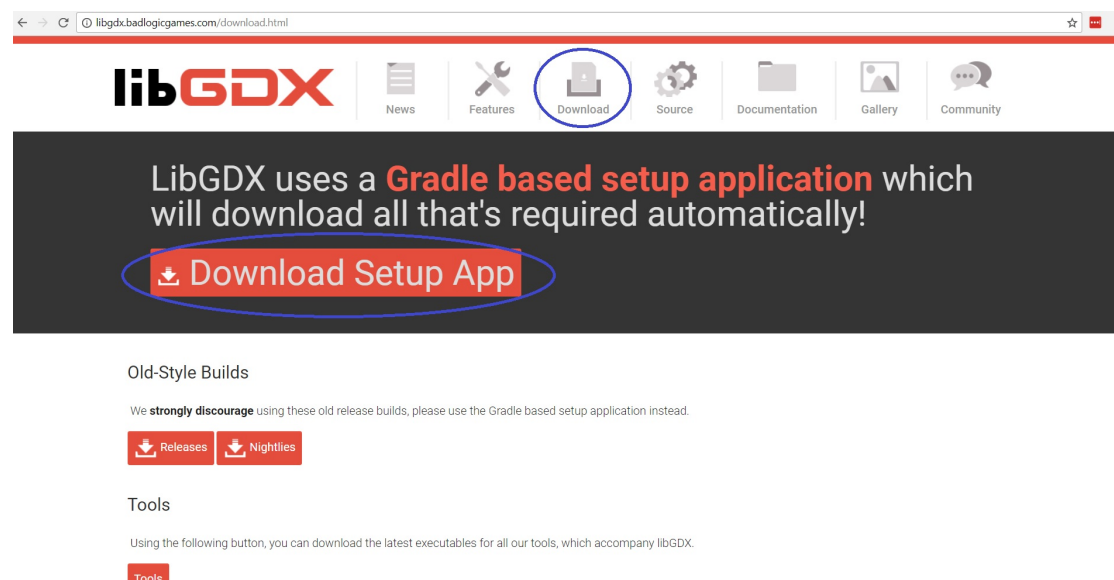
How to set up LibGDX

Image 1.1: How to download LibGDX

1) Go to LibGDX.com and press the "Download"-button (marked in blue in the image above). Afterwards, press the large "Download Setup App"-button.

2) You need Java to open the file you just downloaded. If you don't already have it, download Java 8 SE Runtime Environment for your operative system

3) After you have installed Java, you should be able to run the "gdx-setup.jar"-file that you just downloaded
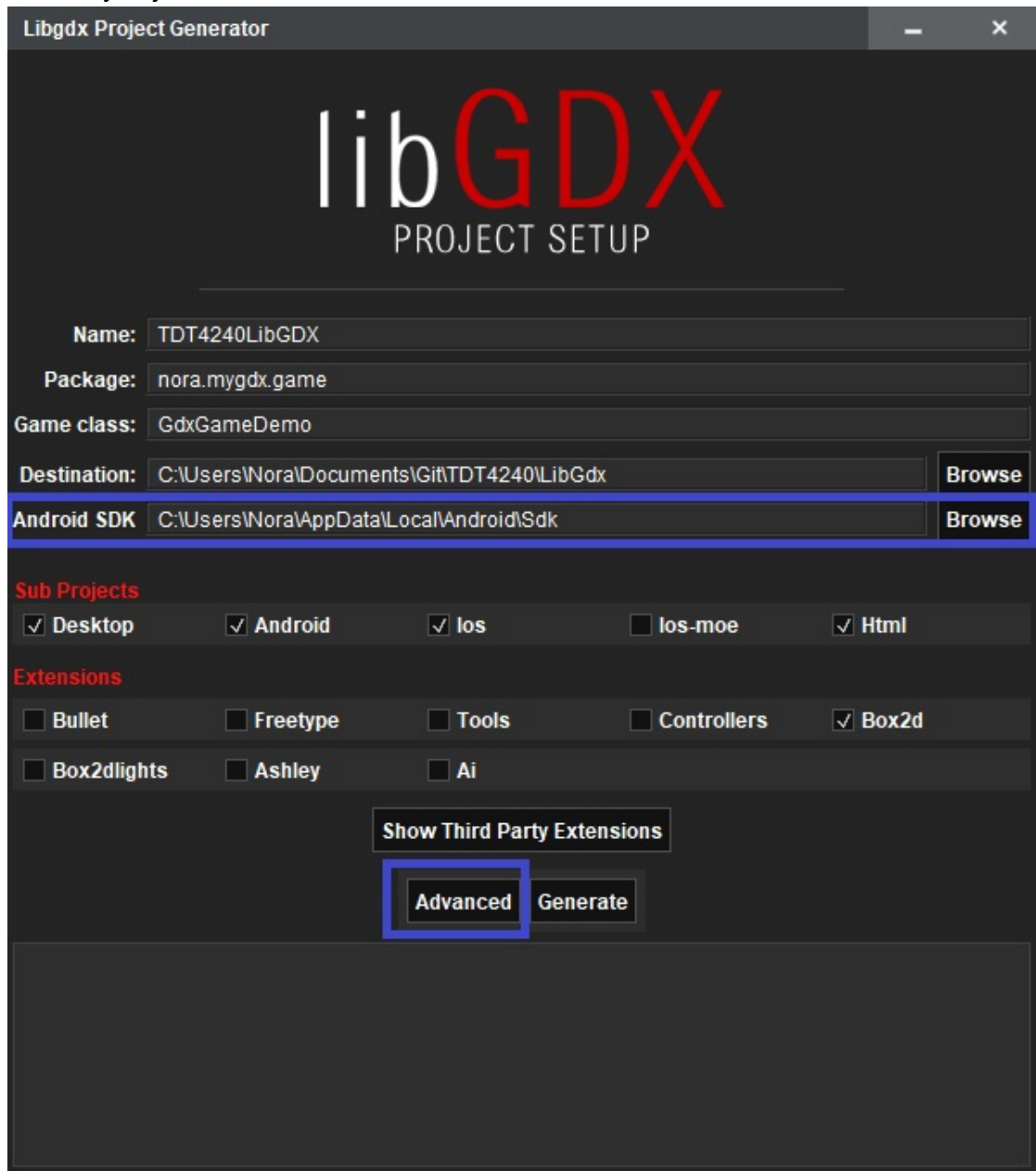

Image 1.2: Shoe the Project Generator where your Android SDK is (Step 4), and press "Advanced" (Step 5)

4) When you run the jar.file you will be prompted to the LibGDX Project Generator-window, here you can choose a project name and etc. You will be asked to show where "Android SDK" is on your computer. Previously when you installed Android Studio you automatically downloaded the SDK as well.

Android SDK is usually (on windows) automatically saved at: *C:\Users\DittBrukernavn\AppData\Local\Android\SDK.* If you are unable to find your SDK, you can download Android SDK only again. *(To do this you have to scroll down to the bottom of the page to: "Get just the command line tools", and download the correct zip-file.)*

5) Press "Advanced" and mark "IDEA" if you are using Android Studio. This is so that LibGDX can generate the right files for your IDEA.
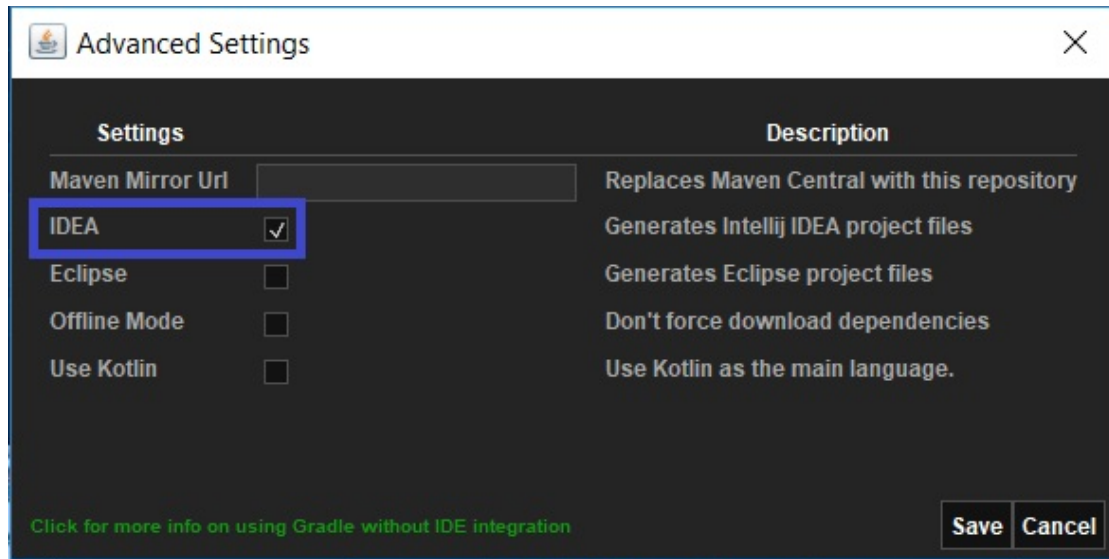

Image 1.3: Choose "IDEA" for the right files to be generated

6) When you press "Generate" you will be told that: "You have a more recent version of android build tools than the recommended. Do you want to use your more recent version?".

Press "Yes", and "Yes" again on the other message.

7) You will see the text "BUILD SUCCESSFUL" when the generating of the files is finished.

8) Now you have to open Android Studio again and press: "Import Project (Gradle, Eclipse, ADT, etc.)"
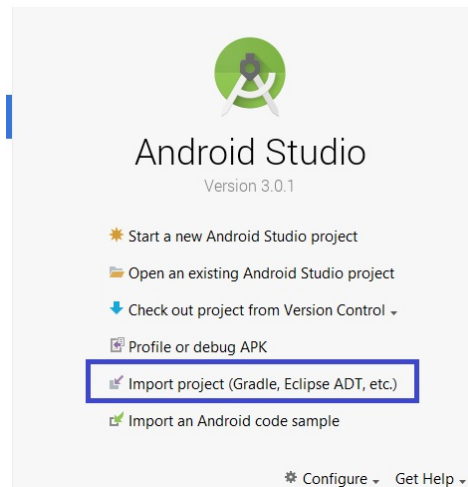

Image 1.4: Import the LibGDX project into Android Studio again

9) Navigate to where you saved the LibGDX-project and press "ok"

10) In the Android world updates happen frequently, and LibGDX is not always synchronized with the latest version of Gradle. Your LibGDX project will not run if your Gradle-version is too far ahead. You have to change the Gradle-version to a version that LibGDX is compatible with. In this case it will be Gradle version 3.3

Press File -> Project Structure, and then choose project "Project" (See Image 1.5 below)
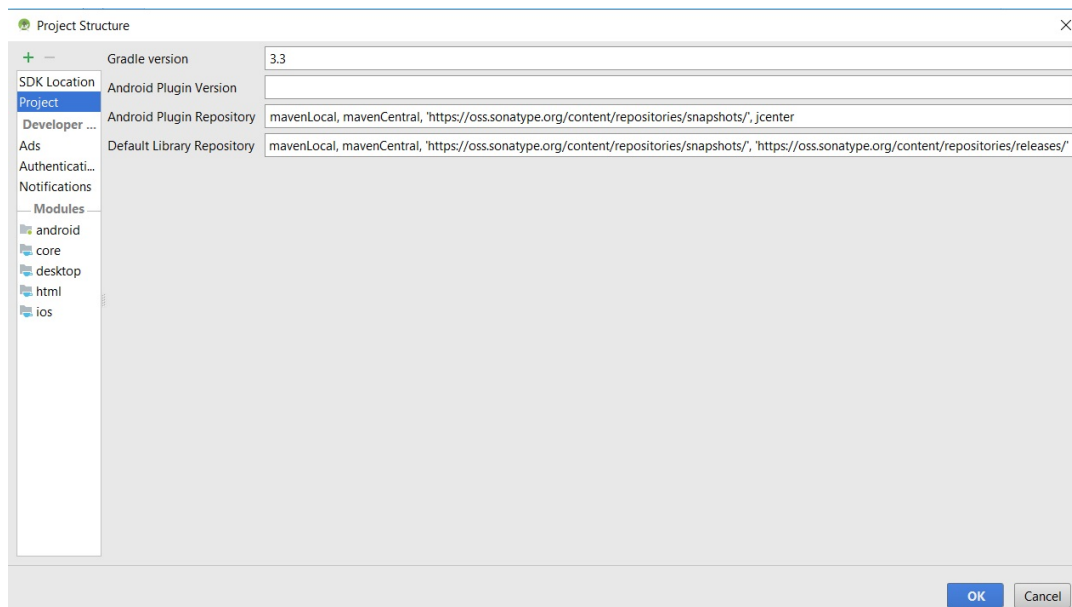

Image 1.5: In "Project" you set "Gradle Version" tl "3.3"

Here you can set the Gradle version to 3.3. Afterwards, you should be able to run the project on an Android Emulator.

11) This is not an obligatory step, but it can pay off to not run the project in Android Emulator. This is because the Android Emulator often uses a lot of RAM, and takes some time to run. Because LibGDX is multi-platform, you can rather run it in the "Desktop"-version.

Press "Run" and "Edit Configurations…"

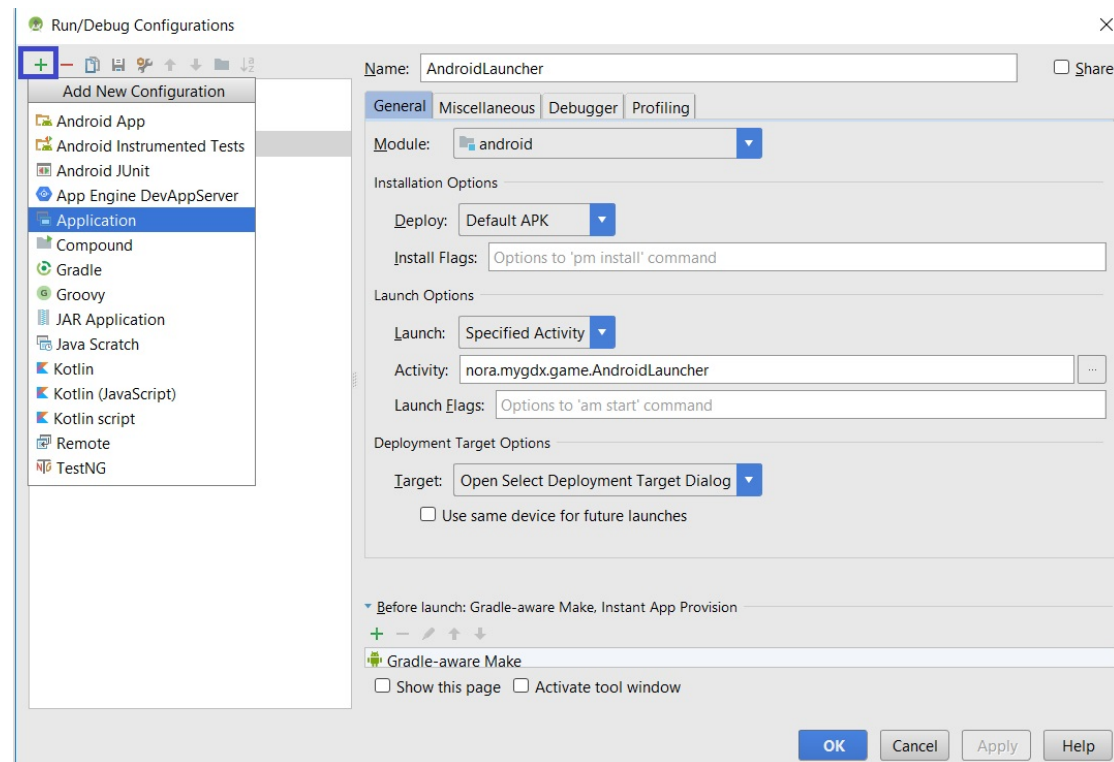Afterwards, press the green plus-sign (see Image 1.6 below), and choose "Application"

Image 1.6: Press + and choose "Application" to make a Desktop-version that you can run

See Image 1.7 for details on what you can write when you are creating the Desktop-launcher. You can set "Desktop" as Name, Afterwards you should pick "desktopLauncher" as MainClass. As "Working Directory" you have to pick the assets-file in your project. This is the place where LibGDX will get all the images from. Finally, you should choose desktop under "Use classpath or module"
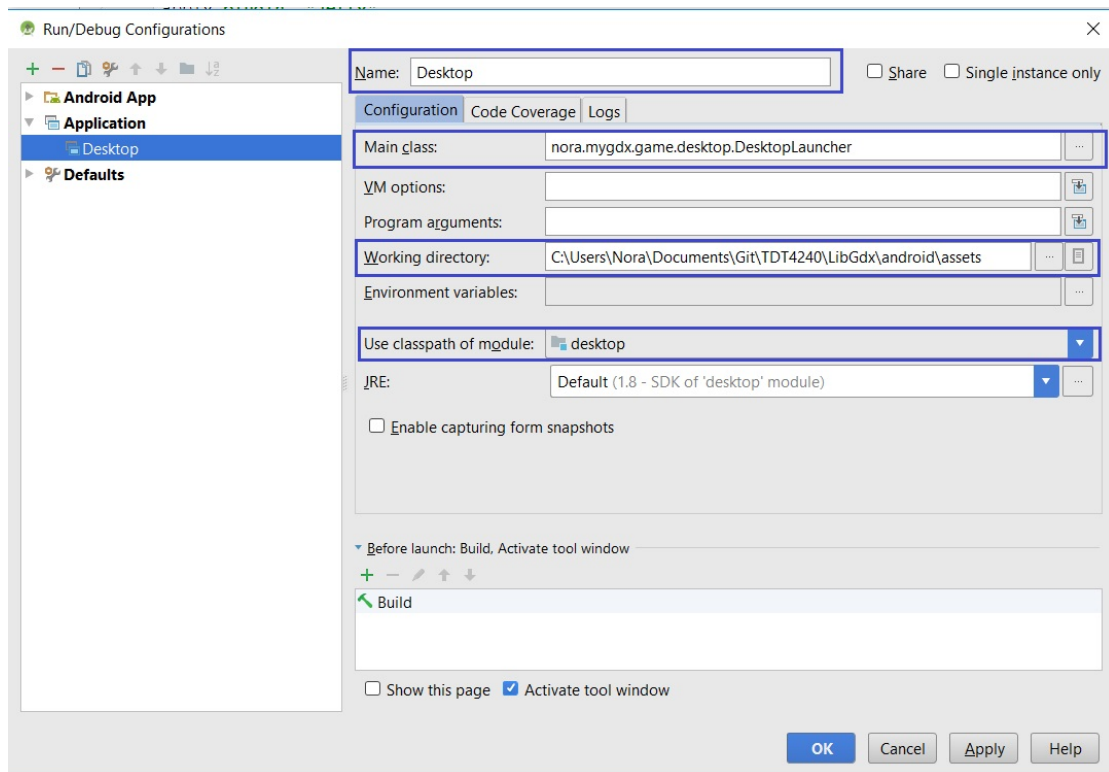
Image 1.7: Everything you have to pick is marked with a blue square

Aftwards you can press "Run" and "Run Desktop" to run the application in Desktop-mode. If LibGDX is used for the final project, it might not be the best idea to *only* run the application in "desktop"-mode. This is because there will be some differences in images/look from desktop mode and Android-mode. Remember to check how your application looks in Android-mode at times, so you don't get a big surprise towards the end of the project. :)