Pre-Project Exercise 1

Technology introduction

The goal of this introduction exercise is to prepare you for the course project, which will account for 60% of your final score, by getting you familiarized with the necessary tools for mobile game development.

1-Technology

The exercise will be done on Android. 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.

- Android Studio (recommended)
- LibGDX framework in Android Studio (recommended if you are a beginner in Android)
- SurfaceView class provided by the android SDK (if you want to develop a native application and to draw game objects)
- ….

But feel free to choose a different approach.

2-Android Studio IDE Guide

This guide shows you how to install Android Studio. Android Studio is the most popular IDE for creating Android Applications. You are free to use another IDE such as Eclipse or IntelliJ, but the setup for Android will then become significantly more difficult.

Step 1 - Download Android Studio

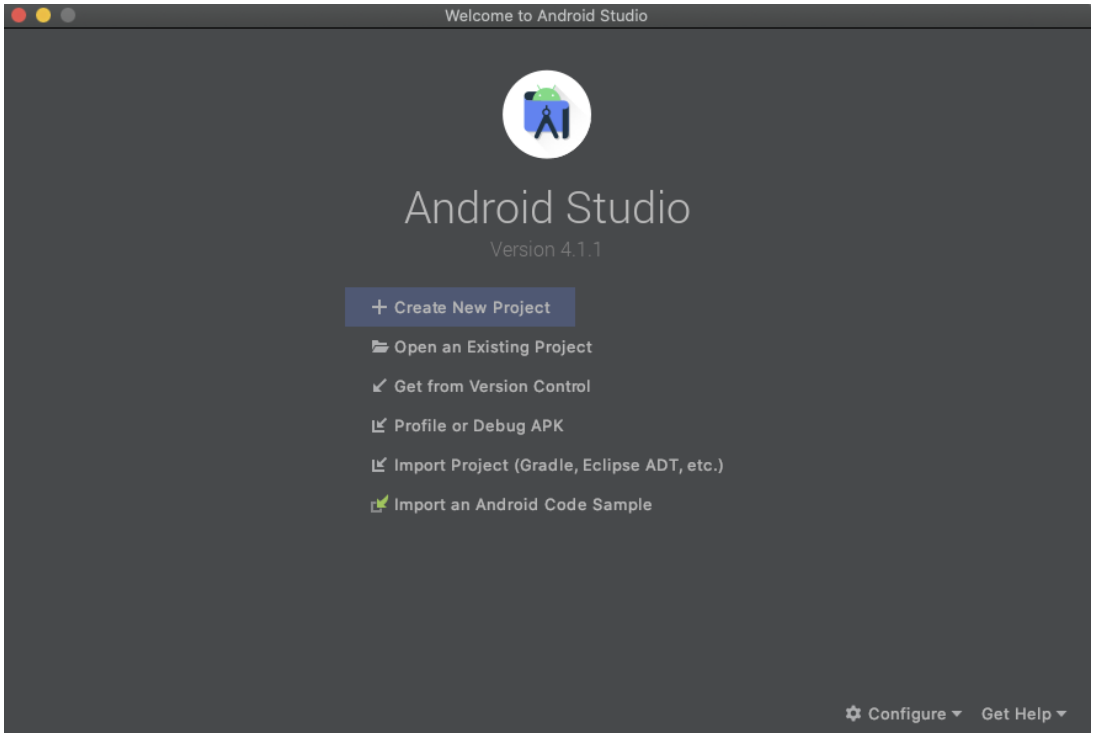
Use the following link to download Android Studio:

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 2 - New Android Project

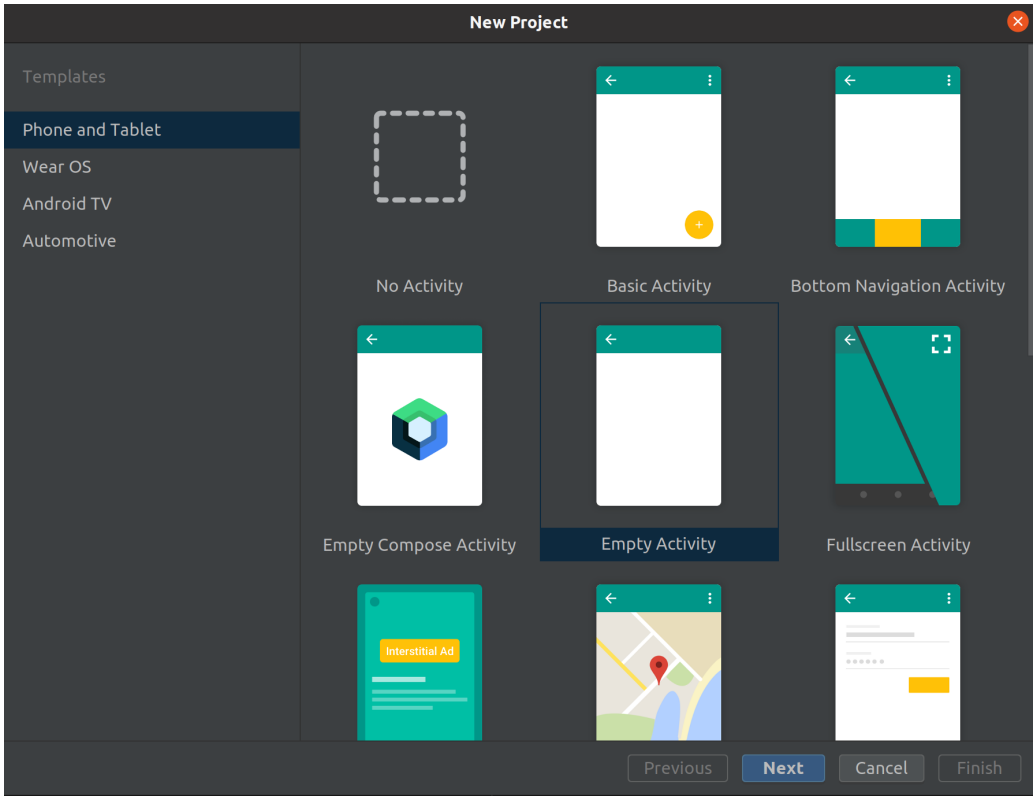
When the Setup Wizard finishes press: “**Create New Project**”.



(If you already have a project open, from the main menu select **File | New | Project**.)

Step 3 - 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 see. For example, a classical “log in”-window.

Choose “Basic Activity” this time. This will lead you to the “Configure Activity”-window, here you can choose the programming language of the application, the rest don’t need to be changed.

You will be prompted to a window where you can write the name of your app, package, choose a Project Location, programming language and minimum SDK.

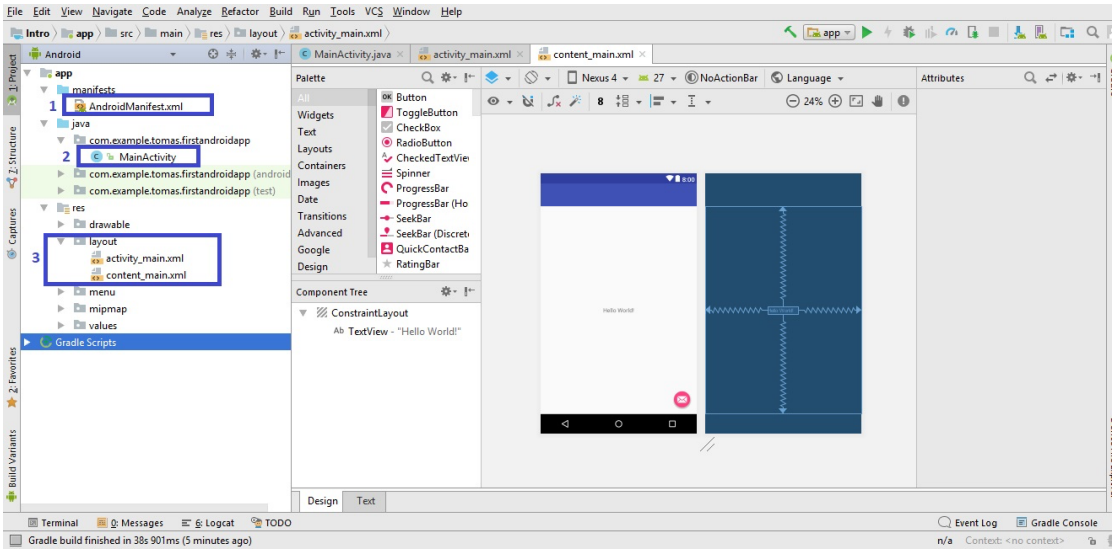
Step 4 - Fix potential errors

When you have created a project, and picked an activity, you could get some errors related to missing platform and you should install them.

If you get different (or more) problems, don’t be afraid to ask for help!

Step 5 - 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 (**Tools > AVD Manager > Create Virtual Device**)

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.

3-LibGDX framework Guide

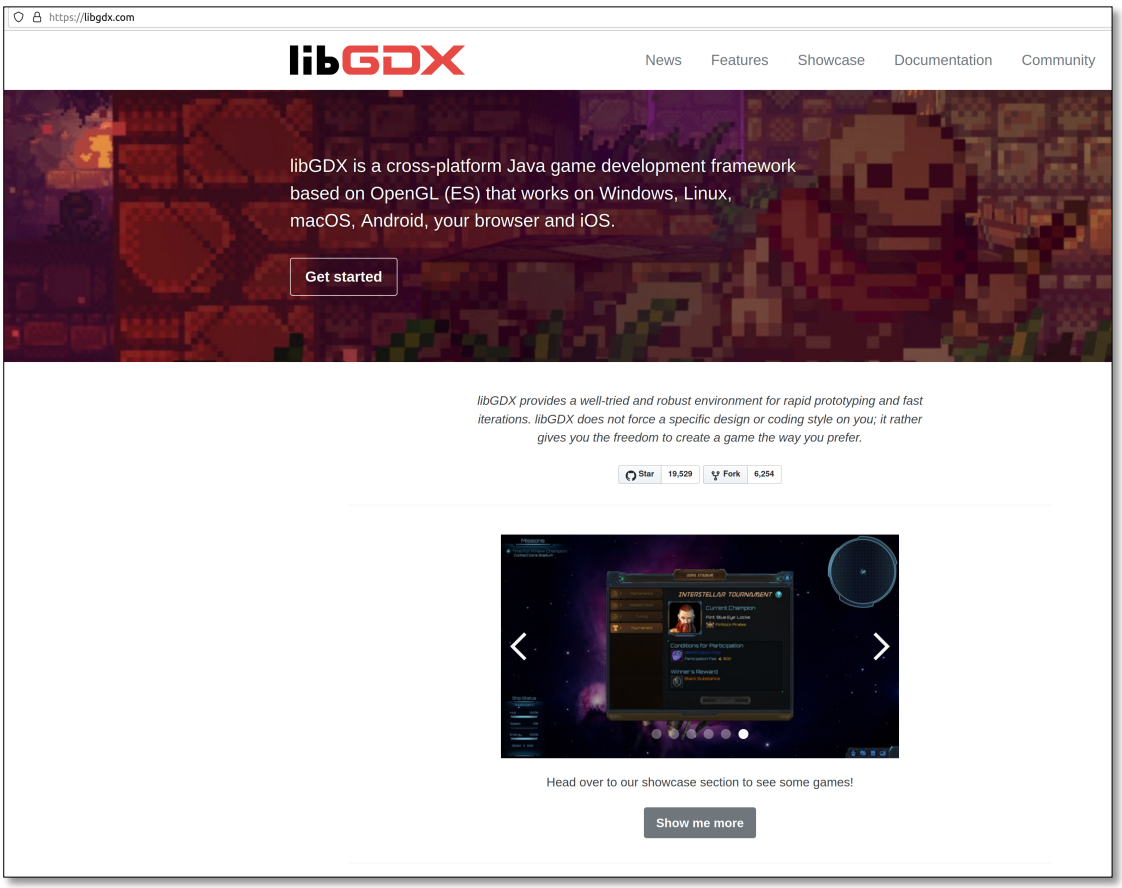
LibGDX offers functionality that can make animations and movement easier to implement. Especially if you are a beginner in Android it can be easier to do this exercise in LibGDX. In addition, if you are making a game with collisions, and movement (such as in the first exercise) for the main project,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

This tutorial-series on YouTube teaches how to make FlappyBird in LibGDX: https://www.youtube.com/watch?v=rzBVTPaUUDg

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

1) Go to LibGDX.com and press the “Documentation”. Afterwards, press the “Generate a Project”-button and Download “Project Setup tool”: https://libgdx.com/wiki/start/project-generation



2) After you have installed Java, you should be able to run the "gdx-setup.jar"-file that you just downloaded. Open your command line tool, go to the download folder and run

Linux:

```
java -jar ./gdx-setup_latest.jar
```

You need Java SE SDK to open/run the file you just downloaded. (for example If you got an error "Exception in thread **"main" java.lang.UnsatisfiedLinkError: Can't load library: /usr/lib/jvm/java-X-openjdk-amd64/lib/libawt_xawt.so"** means you didn't install Java SE SDK.
You should install **Java SE SDK (also called "Java SE Development Kit)**
For example for installing Java SE 17 SDK (or Java SE 17 Development Kit) you should download and install "JDK 17"

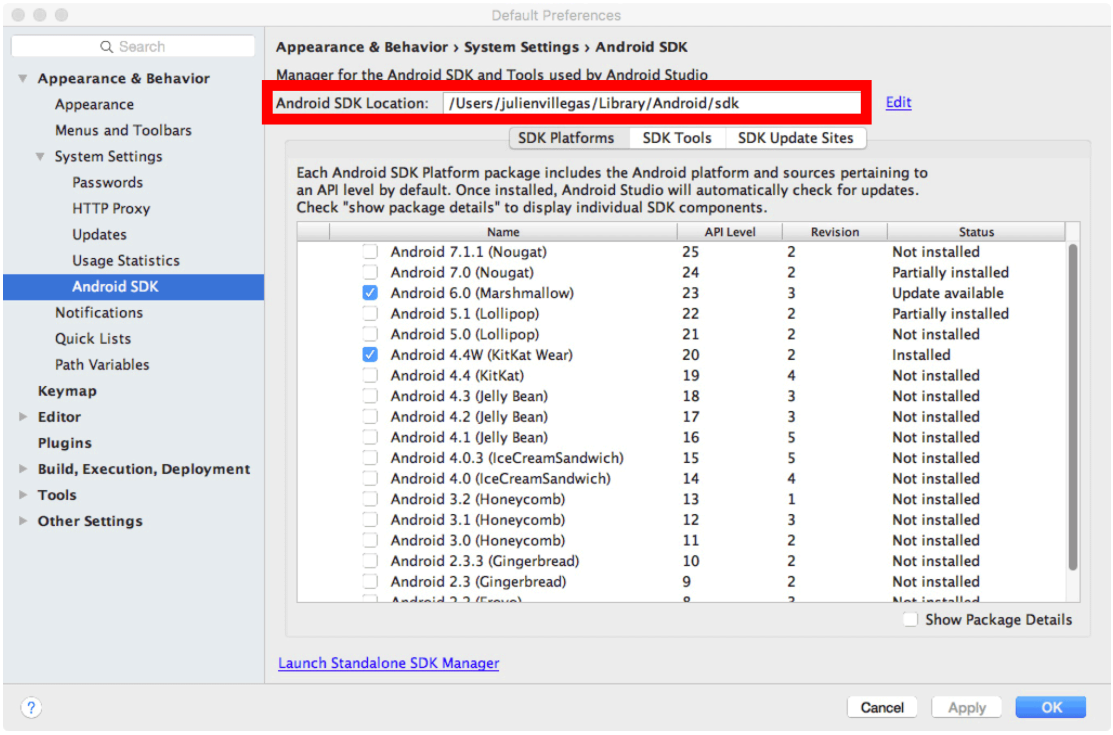
Linux:

```
sudo apt install openjdk-17-jdk
```

If you installed any **Java SE SDK** (Java SE Development Kit), you don't need to install JRE (Java SE Runtime Environment) separately, because JRE is a part of Java SDK.

3) You are asked to provide the following parameters:

Name: the name of the application; lower case with minuses is usually a good idea, e.g. my-game
Package: the Java package under which your code will reside, e.g. com.badlogic.mygame
Game Class: the name of the main game Java class of your app, e.g. MyGame
Destination: the folder where your app will be created
Android SDK: the location of your Android SDK. With Android Studio, to find out where it is, start Android Studio and click
"File/Setting/Appearance & Behavior/System Setting/AndroidSDK/Android SKDK Location".
By default it is in /Users/username/Library/Android/sdk



4) Optional: If you want to code the exercise in Kotlin instead of Java, in the "Advanced" and select "Use Kotlin"
5) Click on "generate"

6) Now you have to open Android Studio again and press file and open the project from destination address, for example

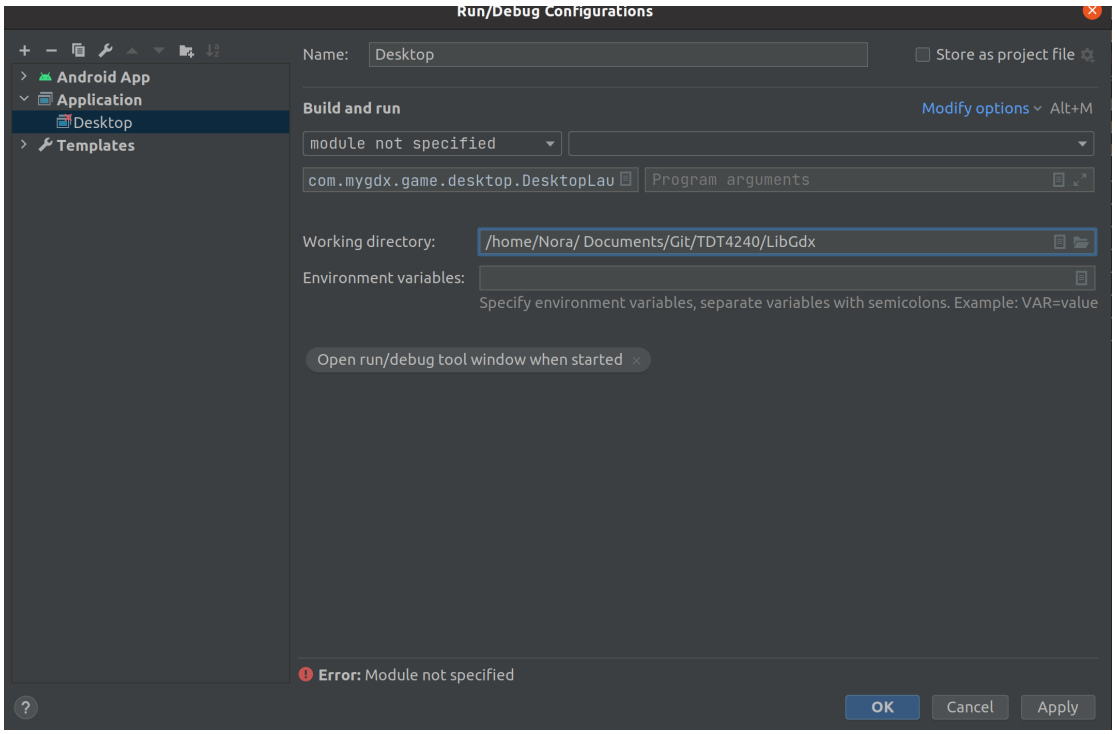
Open “build.gradle”

7) 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 Ctrl+Shift+A, Type **Edit Configurations,** Click **Edit Configurations**
Afterwards, press the green plus-sign, and choose "Application"

Name: Desktop
Main Class: com.mygdx.game.desktop.DesktopLauncher
Working Directory: your LibGDX destination address
For example: C:\users\Nora\Documents\Git\TDT4240\LibGdx
or /home/Nora/ Documents/Git/TDT4240/LibGdx

Press OK



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. :)

4-Exercise 1 Tasks

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

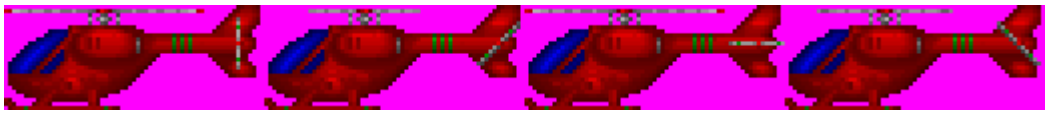
Reuse the helicopter sprite from task 1.

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).

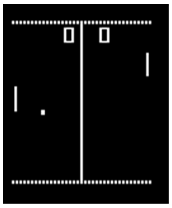
- The ball should start off in the center of the screen and bounce off the paddles, and the upper and lower edges of the screen.

- 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



5-Exercise 1 Delivery

You should deliver all of tasks codes (e.g, Task1.zip, Task2.zip, Task3.zip, Task4.zip) together with either an apk files (e.g, Task1.apk, Task2.apk, Task3.apk and Task4.apk) or a runnable jar file for LibGDX (e.g, Task1.Jar, Task2.Jar, Task3.Jar and Task4.Jar) in an zip file.

Good Luck:)