

## WORD CONNECT HELP DOCUMENTATION

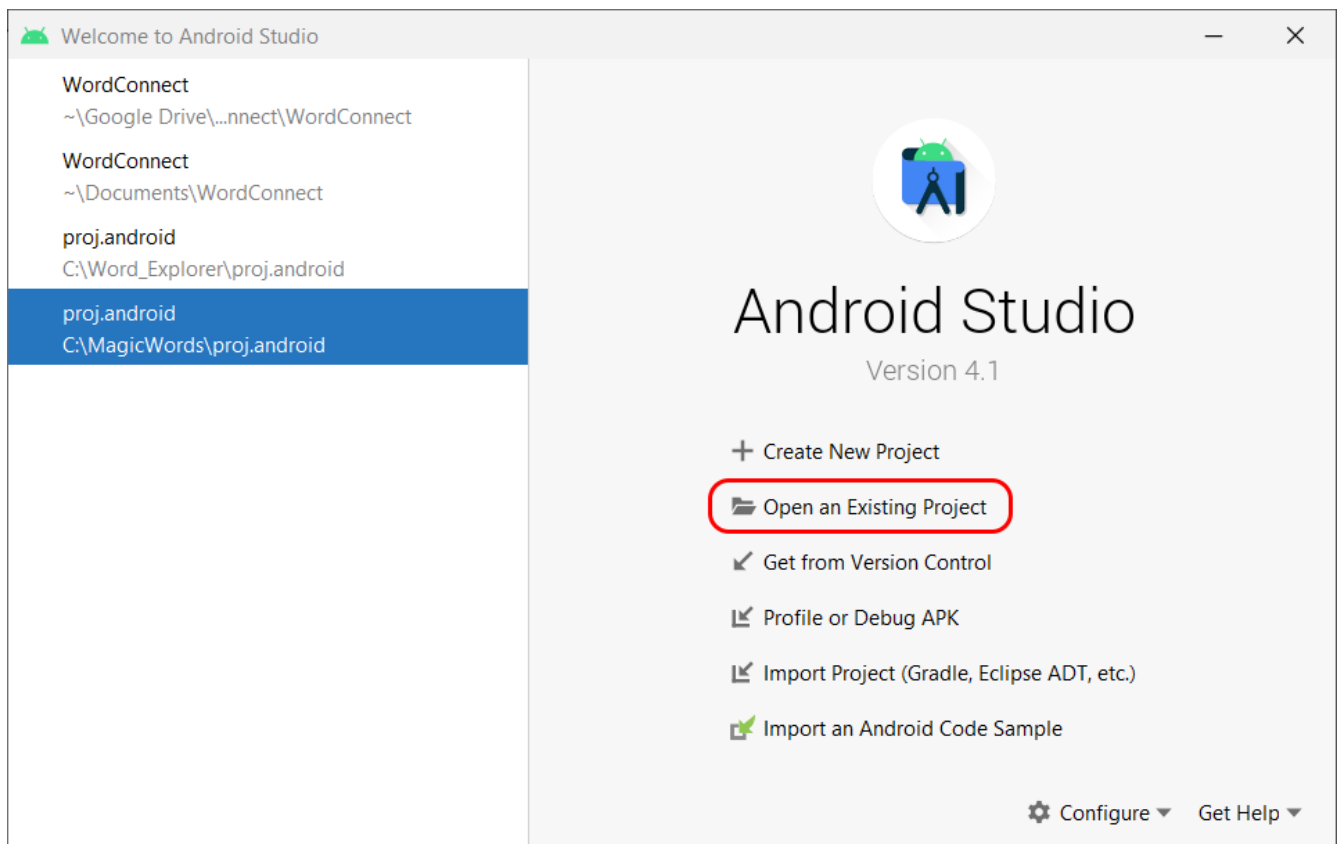
### Warnings

- 1. Ads will not show up until you configure your GDPR settings. Please read this document to see how to configure it.**
- 2. If you don't enter IAP items to Google Play Developer Console the app will crash when you hit the plus button in the app. First enter the items in the order specified in this document and/or strings.xml file. Also don't change the supplied item ids.**

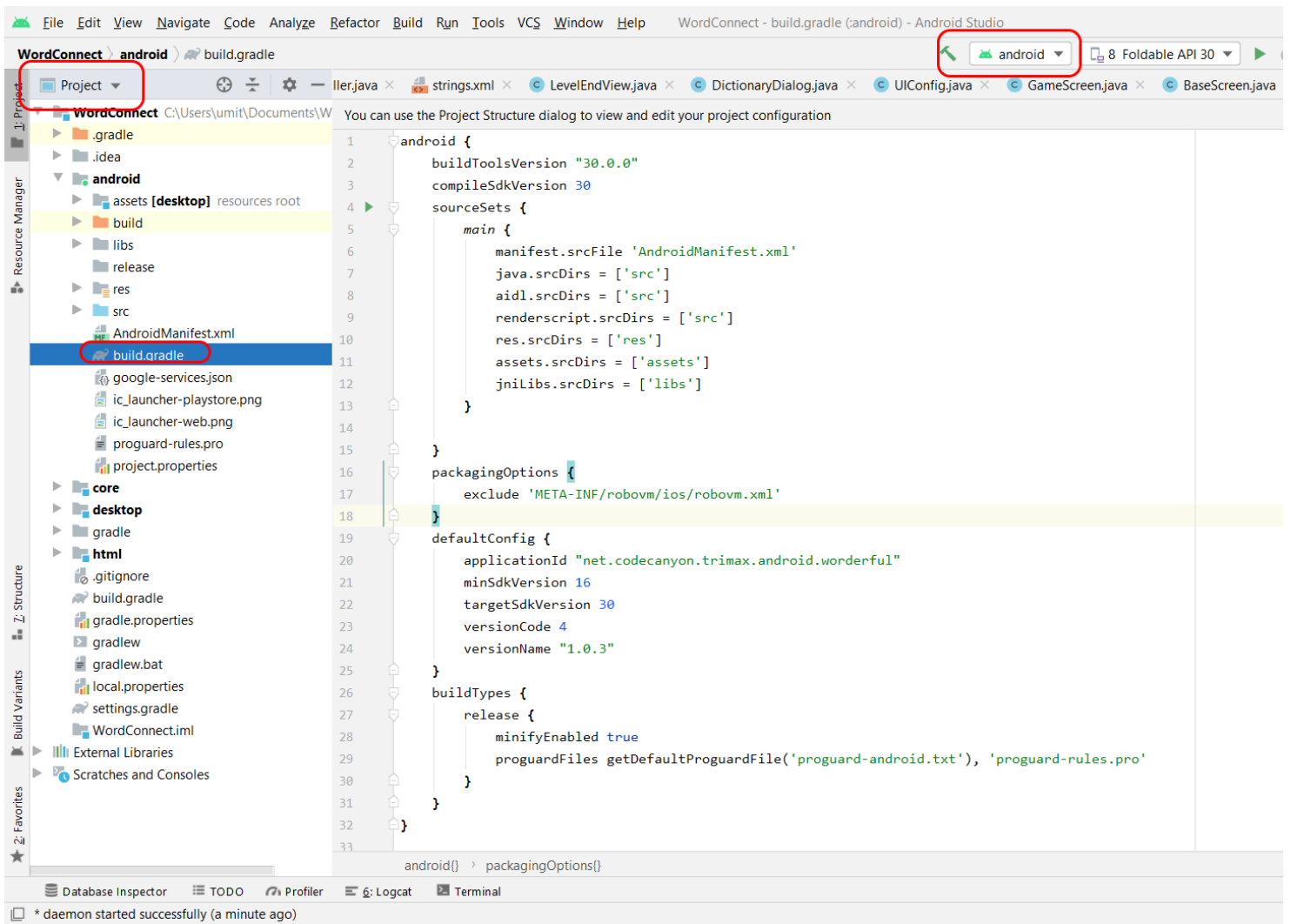
Thanks for purchasing the Word Connect Android Game. Please note that you must make some changes in the game otherwise Google may remove your app and account from Play Store due to repetitive content. Your game should look and feel different to other people's game who have purchased the game previously. Therefore, if you don't make any changes you risk both your self and others. The details of such modifications will be listed in the following sections.

### Setting up the Project

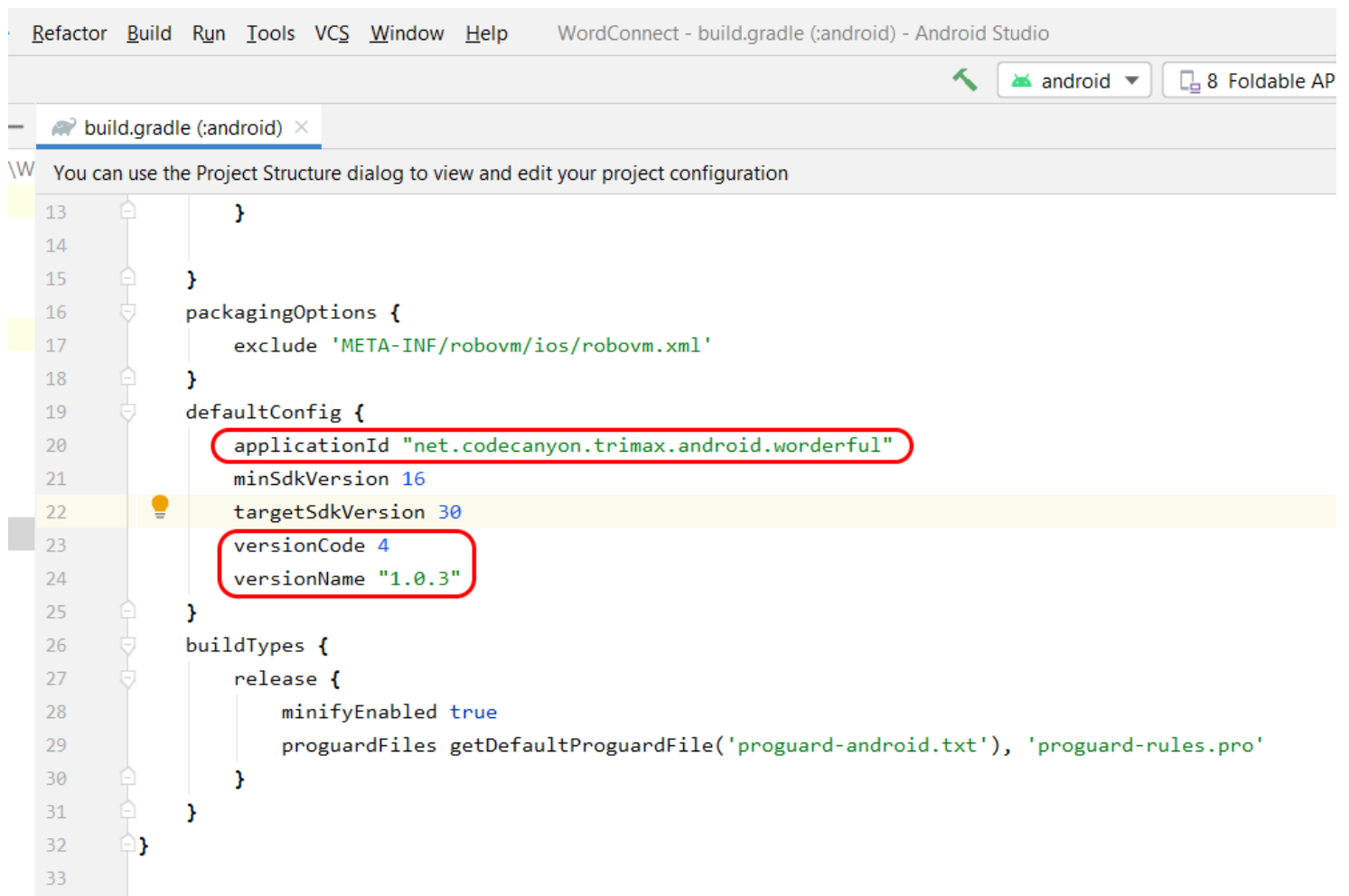
1. Unzip the project you downloaded from Codecanyon.
2. Open Android Studio. Close any existing projects and open the word connect project named WordConnect in the unzipped folder:



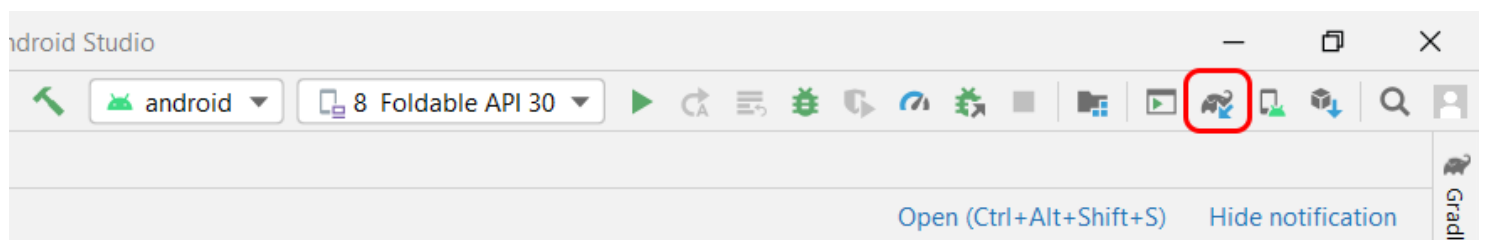
3. Wait for a while until the sync process ends. Make sure Project and Android are selected as shown below and open build.gradle:



4. In this file change the **applicationId**, **versionCode** and **versionName** (shown below). applicationId uniquely identifies your app at Play Store. You can replace the word trimax with your Codecanyon username and it will be unique. Make versionCode 1 and versionName 1.0.0:



5. Android Studio will prompt you to sync the project by popping a yellow balloon on the right-top corner of the screen. If it doesn't you should manually sync it by clicking the button shown below:



If you get the following error:

Execution failed for task ':android:processDebugGoogleServices'.  
No matching client found for package name

open the google-services.json file in the android folder and update the "package\_name" property with your new package id. This is a temporary solution you will completely

replace this json file later on. If changing the package name in this doesn't correct the error, then go to Google Firebase Console, create a project, download your json file and overwrite google-services.json. This procedure is explained in the push messaging section in one of the subsequent pages.

6. Now connect a device or configure an emulator and run the app by clicking the green button:



## Configuring the Game

There is a wide variety of configuration options in the project. Most of the options are commented in source files.

1. `res/values/strings.xml`. This file contains the settings related to Android such as game title, Admob and IAP. Choose a different title for your app other than "Word Connect" because this name is used by many other games (or you can leave it as is, it is up to you). You need to create a rewarded video and an interstitial at Google Admob console. For this go to <https://apps.admob.com/v2/home>. Choose Apps->ADD APP and follow the instructions. Copy and paste all the generated ids into necessary xml tags. Make sure that the value of "ADMOB\_IS\_TESTING\_ADS" is true and enter your real Android device hash id into "ADMOB\_REAL\_DEVICE\_HASH\_ID\_FOR\_TESTING". Otherwise, Google may ban you. But don't forget to make it false before publishing your app to Play Store. **Note that the new Admob API requires you to upload an APK to test ads or you won't see any ads.** I recommend you to upload an APK to internal test track Ads, IAP and other things initially (not production until you finish all the configuration).

Google has also changed setting up **GDPR**. If you are not familiar with the new method, go to Admob home, click the Blocking Controls on the left menu and you will see the "EU user consent" option on the new page. Click this option and configure GDPR. If you don't setup GDPR, Google may terminate your contract. The following steps will enable you to configure GDPR consent:

1. Open [apps.admob.com](https://apps.admob.com)

2. Open your app in app section on left hand side
3. In middle on left hand side there will be option "Blocking Control"
4. Now on a screen click on " Manage EU user consent "
5. Now click on an option. Create and " manage consent for GDPR "
6. Manage consent on Funding Choices.
7. Click on your app
8. Publish your consent

If you enable IAP do not change anything in strings.xml. You should only enter these ids, cost and titles in Google Play Developer Console. For this, go to the console and upload an APK. After your APK is reviewed by Google, at console, select your app, select Monetize->Products->In-app products on the left menu. Now, using the ids (such as 01, 02), enter your IAP items with the correct order. What is the correct order? You should start with the bottom-most item in strings.xml, i.e. 06 and finish with 01. Yes but what do these items correspond to? These ids on their own are useless, so, go to Android Studio, in the project pane, open core->src->word.game->config->GameConfig. In this file search for the function named "setUpIAPToItemMapping". Here you will find the necessary mapping and details. Review these items and decide for a title and cost. You can use the similar titles and pricing as my demo app. When you are finished with it you should test it and purchase items for free without any charge. In the new Google Play Developer Console, go to All apps where all your app titles are seen not specifically Word Connect. On the left menu choose Settings->License testing.

← → ↻ play.google.com/console/u/0/developers/5534403249647247790/license-tester

Apps ⌚ 🖼️ 🍏 📁 anims 📺 instantreality 1.0 - t... 📁 opengl 🔄 glfm/src at master ... 🌈 ColorSpace - Color...

☰ Google Play Console 🔍 Search Play Console

All apps 1

Inbox 6

Users and permissions

Order management

Download reports

Settings

Developer account

Preferences

Email lists

License testing 2

Manage game projects

Pricing templates

## License testing

Test your licensing and in-app billing integrations. [Show more](#)

Add license testers

License testers ?

License response

3 user1@example.com,

Add 1 or more email addresses with a Google account.

RESPOND\_NORMALLY

License testers will get this apps that haven't been uplo

On this page add a Google account email address who will do the testing and save changes. Note that it may take a few hours to take effect. We are not finished yet. Now go to All apps and select word connect. On the new page click Testing->Internal testing. Add the same person's account here as well and copy the app link towards the bottom of the page:

The screenshot shows the Google Play Console interface for internal testing. On the left sidebar, the 'Testing' menu item is highlighted with a red box and the number 1. Below it, the 'Internal testing' option is also highlighted with a red box and the number 2. A red arrow with the number 4 points to the 'Feedback URL or email address' field. On the right, the 'Testers' section is highlighted with a red box and the number 3. It contains a table with two rows: 'List name' and 'Users'. The 'List name' row has a checkbox checked, and the 'Users' row has a value of 1. Below the table, there is a 'Create email list' link and a text input field containing 'mahmutacar420@gmail.com'.

Google Play Console

Search Play Console

Internal testing

Create and manage internal testing releases to make your app available to up to 100 internal testers. [Learn more](#)

Track summary

Active • Latest release: 1.0.1

Releases Testers

Testers

Up to 100 testers can join your internal tests. You can choose more than 100 testers, but only the first 100 to join will be successful.

Testers
<input checked="" type="checkbox"/> List name
<input checked="" type="checkbox"/> Users

1

2

3

4

Create email list

mahmutacar420@gmail.com

Let testers know how to provide you with feedback

The link contains the path to your private-internal word connect game if you uploaded your app to the internal test track. Send this link to the test user via email. He/she must open it and agree with the testing terms and conditions of Google Play. Otherwise, he won't be able to find the app in Google Play Store App.

In the test user's device, open the app check if "Remove Ads" item is at the top and small coins items are at the bottom and bundles are in the middle of these. If not, then the ordering of the IAP items on the Play Console is wrong. To correct this, go to Android Studio and open `android/src/main/java/com/example/wordconnect/activities/IAPActivity`. In this file, search for the line that contains `Collections.reverse(list);`. Disable this line by putting 2 slashes at the beginning so that it is like this:

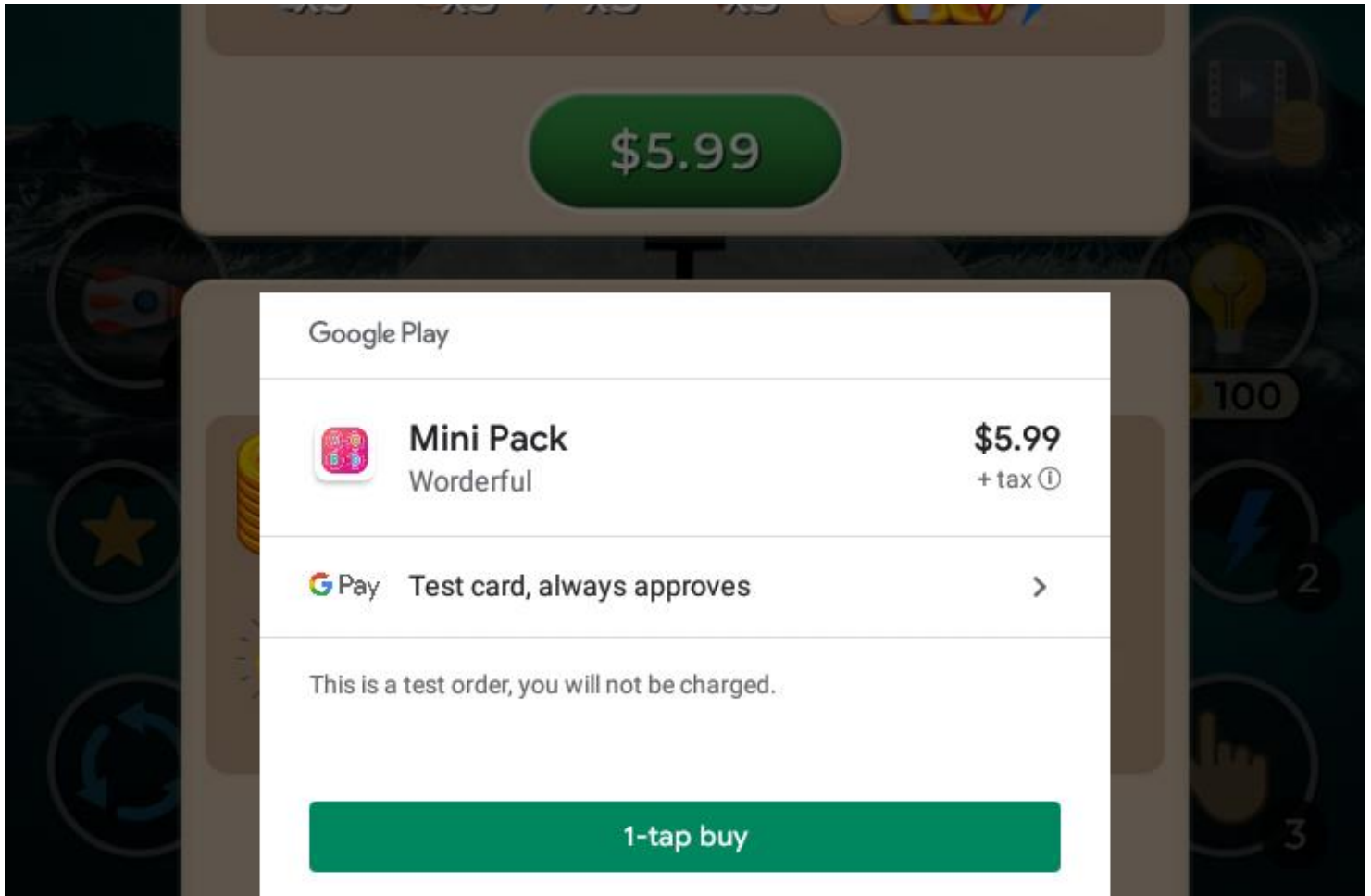
```
//Collections.reverse(list);
```

When a user opens the shopping dialog the items titles are appended with the application's name. Google sends such a title even if you don't want it this way. To display the exact title you want, please edit the `strings.properties` files in `android/assets/data/en` or `tr` folders.

It won't reverse the items any longer. Now, it is time to make a fake purchase. Beware that if you purchase the remove ads, you won't be able to see it again because it is a one time purchase and will be missing forever. Now, to verify that the purchase you make is



free of charge, the purchase card that appears just before the transaction must be similar to the one below, i.e. it must say "This is a test order, you will not be charged".



If yours is different, then review the previous steps and make sure 6-7 hours have elapsed after you uploaded your app and assigned someone to test it.

The last option in the strings.xml is the support email. Type in your email address so that your users can ask you questions.

2. core/src/word/game/config/GameConfig. This is another settings file. It encapsulates the settings related to the game itself. It is easy to change the settings. If you don't have any programming experience don't worry, they are simple. true/false mean enabled/disabled. Some numerical values may have decimal places, in such cases just don't to append the number with f. Yes, the letter f in alphabet. That's all there is to it.

Note that after making changes please test it in the game to see if it works fine and is convenient.

3. `core/src/word/game/config/UIConfig`. This file controls the look and feel of the game. There are numerous color, fonts and size options to change. Before changing any values don't forget the information at the top of the page. If making configuration in this file doesn't satisfy you for skinning, I will later explain how to replace the graphics.

4. `core/src/word/game/config/SoundConfig`. This file is relatively less important. It specifies the volume level of each sound effect.

## Changing Graphics

You must change at least the game logo, icons and background images of the game. You can get free graphics and background images at:

- <https://www.pexels.com/>
- <https://pngtree.com/>
- <https://www.freepik.com/>

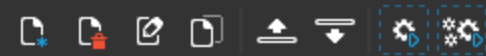
Please note that the tools required for editing sprite sheets and fonts require Java installed on your computer. The graphics of the game are in the `sprite_sheets` folder in the zip file. Each folder named `atlas_1`, `atlas_2`... correspond to a sprite sheet, i.e. the contents of each folder are fused into a single image for efficiency. The tool that packs these images are free of charge. You can download it from <https://github.com/crashinvaders/gdx-texture-packer-gui/releases>. As of this writing, the latest version is 4.9.0 but it is not very stable. So I recommend 4.8.2.

Double click the `word_connect.tpproj` file to open the sprite sheets in this program. It is in the same folder where `db`, `font` and `psd` are. Since the location of source image files and destination folder in your pc are different to my computer, you need to update the paths. Below, number 1 (see the next screenshot) specifies all the sprite sheets. Click the first one, select all files in the bottom pane (number 2) and delete them all by clicking number 3. Now, using the number 4, select all the images that belong to sprite sheet `atlas_1` from your computer (`atlas_1` folder). Then choose the destination directory (number 5) which is `WordConnect/android/assets/textures`

Repeat this process for `atlas_2`, `atlas_3` and `atlas_4`.

File Pack Tools Help

Pack list



atlas1\_

atlas2\_

atlas3\_

atlas4\_

1

Pack general

Output dir C:\Users\umit\Google Drive

File name atlas1\_.atlas

5

Pack files



3

- .../sprite\_sheets/atlas\_1/arrow.png
- .../sprite\_sheets/atlas\_1/arrow\_left.png
- .../atlas\_1/arrow\_right.png
- .../atlas\_1/back\_down.png
- .../sprite\_sheets/atlas\_1/back\_up.png
- .../sprite\_sheets/atlas\_1/bg\_green.png
- .../sprite\_sheets/atlas\_1/bg\_red.png
- 9 .../atlas\_1/btn\_dialog\_disabled.9.png
- 9 .../atlas\_1/btn\_dialog\_down.9.png
- 9 .../atlas\_1/btn\_dialog\_up.9.png
- .../sprite\_sheets/atlas\_1/bundle1.png
- .../sprite\_sheets/atlas\_1/bundle2.png
- .../sprite\_sheets/atlas\_1/bundle3.png
- .../sprite\_sheets/atlas\_1/bundle4.png
- .../atlas\_1/bundle\_coins.png
- .../atlas\_1/bundle\_finger\_hint.png
- .../bundle\_multi\_random\_hint.png
- .../atlas\_1/bundle\_random\_hint.png
- .../atlas\_1/bundle\_rocket.png
- .../atlas\_1/calendar\_cell\_checked.png
- .../sprite\_sheets/atlas\_1/close.png
- .../atlas\_1/close\_button\_disabled.png
- .../atlas\_1/close\_button\_down.png
- .../atlas\_1/close\_button\_up.png

2

Global settings

File type PNG

Encoding RGBA8888

Compression None

Pack settings

Min page width 16

Min page height 16

Max page width 2048

Max page height 2048

Alpha threshold 0

Min filter Linear

Mag filter Linear

Padding X 2

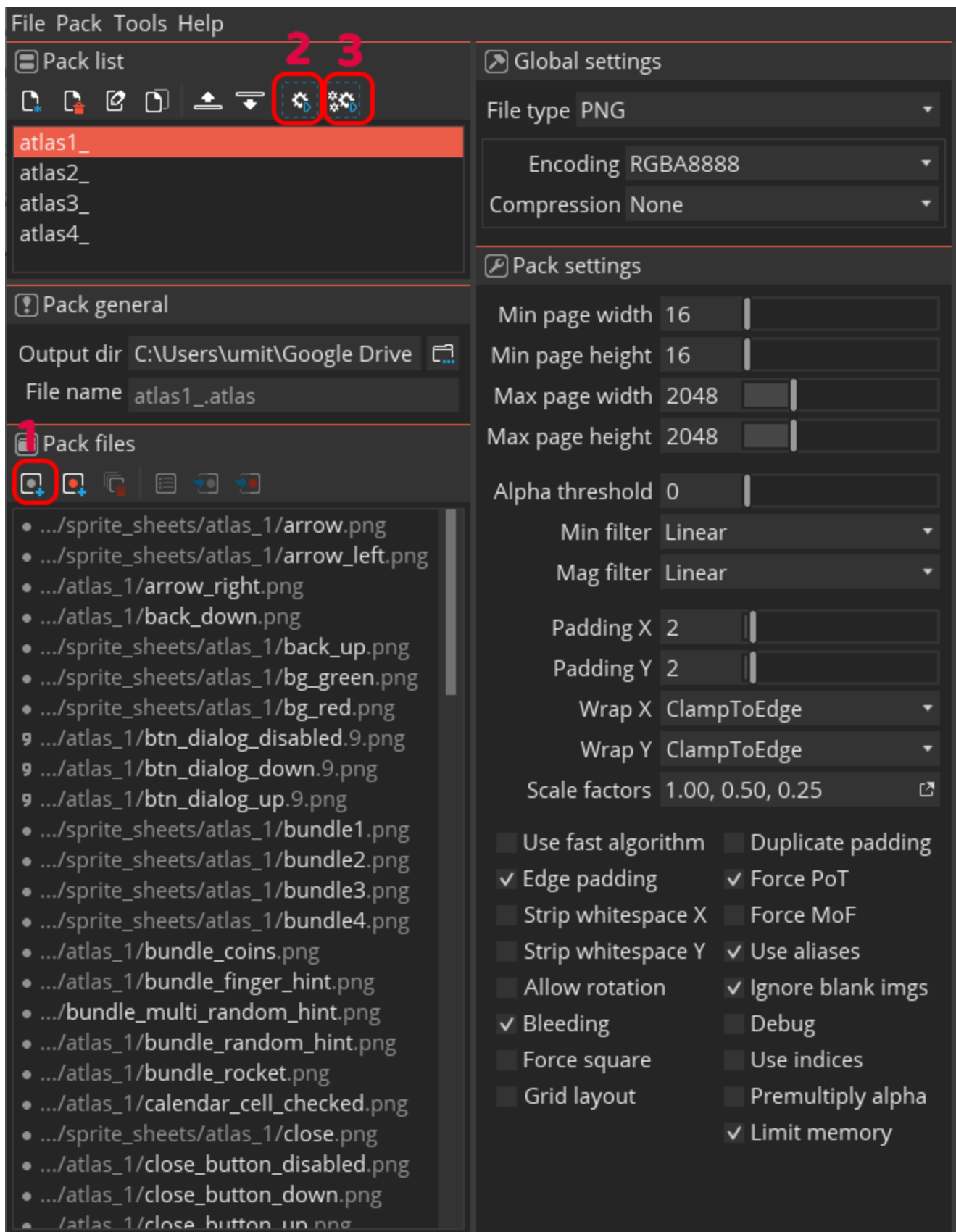
Padding Y 2

Wrap X ClampToEdge

Wrap Y ClampToEdge

Scale factors 1.00, 0.50, 0.25

- ☐ Use fast algorithm
- ☒ Edge padding
- ☐ Strip whitespace X
- ☐ Strip whitespace Y
- ☐ Allow rotation
- ☒ Bleeding
- ☐ Force square
- ☐ Grid layout
- ☐ Duplicate padding
- ☒ Force PoT
- ☐ Force MoF
- ☒ Use aliases
- ☒ Ignore blank imgs
- ☐ Debug
- ☐ Use indices
- ☐ Premultiply alpha
- ☒ Limit memory



You replace an existing image by overwriting it. If you want to replace a scale 9 image ([https://en.wikipedia.org/wiki/9-slice\\_scaling#:~:text=9%2Dslice%20scaling%20\(also%20known,a%20grid%20of%20nine%20parts.\)](https://en.wikipedia.org/wiki/9-slice_scaling#:~:text=9%2Dslice%20scaling%20(also%20known,a%20grid%20of%20nine%20parts.))), create your graphic, export it from Photoshop (or what ever you are using) to a temporary location on your disk. You need to specify the regions that won't scale in this file. To do this go to: <https://romannurik.github.io/AndroidAssetStudio/nine-patches.html#&sourceDensity=640&name=example>. When you are finished, download the zip and copy the highest resolution image from the zip to the corresponding atlas\_ folder. If you want to add a new image, copy it to the atlas\_4 folder (this sprite sheet has some empty space in it), select the new file clicking number 1 (see the next screenshot) and export the sprite sheet clicking 2. If you click number 3, all the sprite sheets will export which is slow. Don't change any of the settings in the texture packer program:

The game has been designed to run efficiently across all screen sizes and devices. Therefore, each sprite sheet is exported to 3 different scale sizes. The smallest one runs on older small sized screens and the largest one goes to the high-resolution ones. You can see the result of such scaling in the WordConnect/android/assets/textures folder. Have a look at the file names. The images in atlas\_ folders are in 1:1 scale, when exported they are divided into 2 and then 4. This results in 3 different sprite sheets for each folder.

\_hdr-> 1:1  
\_hd -> 1:2  
\_sd -> 1:4

Each sprite sheet png image can be 2048x2048 maximum. The texture packer program creates new pages when the capacity is overflowed. However, these automatically created pages are not supported in the word connect game due to the scaling feature mentioned above. You can see the number of sheets generated for each folder at the top-right part of texture packer. If it exceeds 1 just delete them all. If you run out of space in folder\_4, create a new folder on your hard drive naming it folder\_5, create a new texture pack in the texture packer by clicking number 1 (see the next screenshot). Add your images as instructed in the previous paragraphs. The most important step here is to set the scaling values. Do this by clicking number 2:

File Pack Tools Help

**1** Pack list



atlas1\_

atlas2\_

atlas3\_

atlas4\_

Pack general

Output dir C:\Users\umit\Google Drive

File name atlas1\_.atlas

Pack files



- .../sprite\_sheets/atlas\_1/arrow.png
- .../sprite\_sheets/atlas\_1/arrow\_left.png
- .../atlas\_1/arrow\_right.png
- .../atlas\_1/back\_down.png
- .../sprite\_sheets/atlas\_1/back\_up.png
- .../sprite\_sheets/atlas\_1/bg\_green.png
- .../sprite\_sheets/atlas\_1/bg\_red.png
- 9 .../atlas\_1/btn\_dialog\_disabled.9.png
- 9 .../atlas\_1/btn\_dialog\_down.9.png
- 9 .../atlas\_1/btn\_dialog\_up.9.png
- .../sprite\_sheets/atlas\_1/bundle1.png
- .../sprite\_sheets/atlas\_1/bundle2.png
- .../sprite\_sheets/atlas\_1/bundle3.png
- .../sprite\_sheets/atlas\_1/bundle4.png
- .../atlas\_1/bundle\_coins.png
- .../atlas\_1/bundle\_finger\_hint.png
- .../bundle\_multi\_random\_hint.png
- .../atlas\_1/bundle\_random\_hint.png
- .../atlas\_1/bundle\_rocket.png
- .../atlas\_1/calendar\_cell\_checked.png
- .../sprite\_sheets/atlas\_1/close.png
- .../atlas\_1/close\_button\_disabled.png
- .../atlas\_1/close\_button\_down.png
- .../atlas\_1/close\_button\_up.png

Global settings

File type PNG

Encoding RGBA8888

Compression None

Pack settings

Min page width 16

Min page height 16

Max page width 2048

Max page height 2048

Alpha threshold 0

Min filter Linear

Mag filter Linear

Padding X 2

Padding Y 2

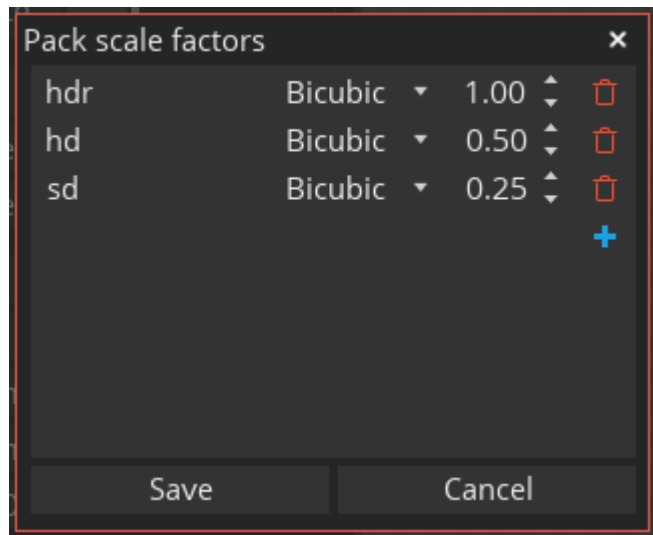
Wrap X ClampToEdge

**2** Wrap Y ClampToEdge

Scale factors 1.00, 0.50, 0.25

- ☐ Use fast algorithm
- ☒ Edge padding
- ☐ Strip whitespace X
- ☐ Strip whitespace Y
- ☐ Allow rotation
- ☒ Bleeding
- ☐ Force square
- ☐ Grid layout
- ☐ Duplicate padding
- ☒ Force PoT
- ☐ Force MoF
- ☒ Use aliases
- ☒ Ignore blank imgs
- ☐ Debug
- ☐ Use indices
- ☐ Premultiply alpha
- ☒ Limit memory

It should look like this:



Finally, specify the destination folder (as instructed before). When you export sprite sheets from the program, sometimes, some of the sheets don't export, showing an error using a cross sign. In such cases, just ignore it and export again. Now that you have some assets for the game, the game doesn't know about these files. This will be explained in one of the following sections.

To save memory, you should shrink the size of your sprite sheets, background and fonts. You can do this at: <https://tinypng.com/>

There is also a PSD folder in the zip file. It contains some of the graphics but not all of them.

## Editing Fonts

The fonts in the game are not TTF because it renders too slow. Instead, bitmap fonts are used. Libgdx has a free font generator called Hiero. Download it from: <https://github.com/libgdx/libgdx/wiki/Hiero>. The font related files are in the font folder (not the fonts folder in android/assets, the other one in the zip file). This folder contains the font projects and TTF files. If you want to add some new characters to an existing font, open the Hiero generator tool. There are a number of font sizes and some have shadows on them. The montserrat\_black series files contain only numbers to show the amount of coins earned after watching a rewarded video, so you don't need to edit these ones. As an example, in Hiero open "montserrat\_semibold\_board\_hd.hiero" from the File menu (the board series show on dial and game board). First, set the correct path to the Montserrat-SemiBold.ttf file by clicking number 1 (see the next screenshot). Then, go to the Sample Text section (number 2) and type or paste the missing characters. Afterwards,

save the updated hiero file by going to File->Save Hiero settings file (it works like save-as rather than save). Finally, export the font files with File->Save BMFont files. Repeat this process for the other hiero font project files. When you are finished with one hiero project file, you should close the generator and reopen it, otherwise, the generator tends to save the current file with name of the previous project. Armed with this knowledge, you can also create new font files or replace the existing ones completely.

## **Adding New Assets to the Game**

You may prefer to add new assets to the game. All the files loaded into the app are listed in core/src/word/game/managers/ResourceManager. Similar files are grouped together and each group is commented. Now, add your files into the suitable group with a unique variable name with the same structure as the others. The files preloaded in the core/src/word/game/screens/SplashScreen. Open this file and find the function name "loadAssets". This function loads the assets that were defined in the ResourceManager file. Depending on what you wish to load, copy an existing line, paste it and replace its file name with the new one. Now the fun part, how will you use the asset in the game:

For an image (aka atlas region) in a sprite sheet:

```
TextureAtlas atlas4 = wordConnectGame.resourceManager.get(ResourceManager.ATLAS_4, TextureAtlas.class);
TextureAtlas.AtlasRegion myRegion = atlas4.findRegion("file_name_of_the image");
```

```
Image myNewImage = new Image(myRegion);
addActor(myNewImage);
```

For font:

```
BitmapFont myNewFont = wordConnectGame.resourceManager.get(ResourceManager.myNewFont, BitmapFont.class);
```

For sound effect:

```
if(!ConfigProcessor.muted) {
    Sound sound = wordConnectGame.resourceManager.get(ResourceManager.SFX_NEW, Sound.class);
    sound.play();
}
```

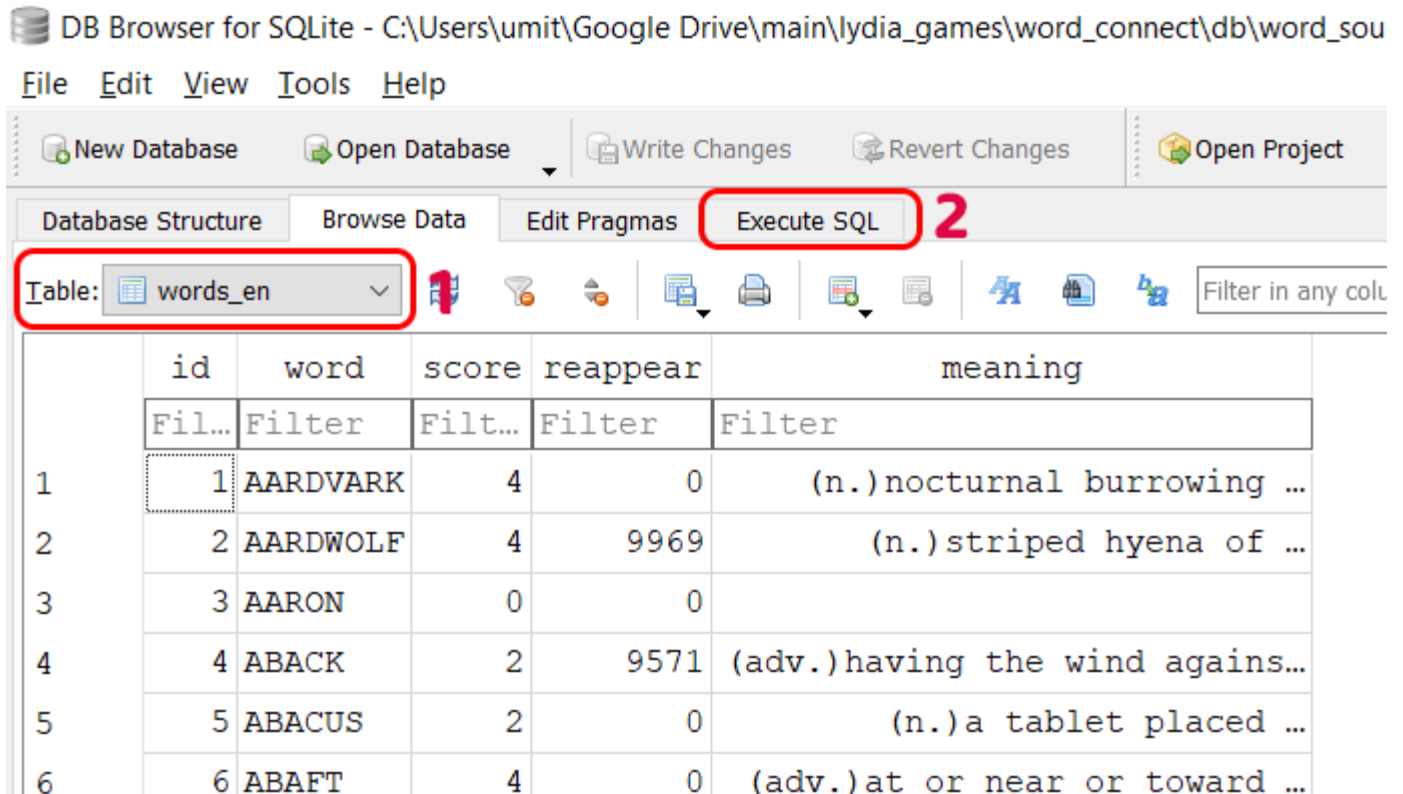
## **Generating New Levels (And Adding a New Language)**

New Levels are generated at a separate Java program that runs in Eclipse IDE. You may wish to add a new language to the game or want to generate all the levels of existing English and Turkish from scratch (so that they will look different to the other ones.). If your case is one of these, keep reading. But, creating ten thousand levels may take 1 to 4 days depending on the speed of your computer.

1. First create a folder in android/assets/data and name it as the language code of your language such as "es" or "de". To generate levels you need words. Where will you put

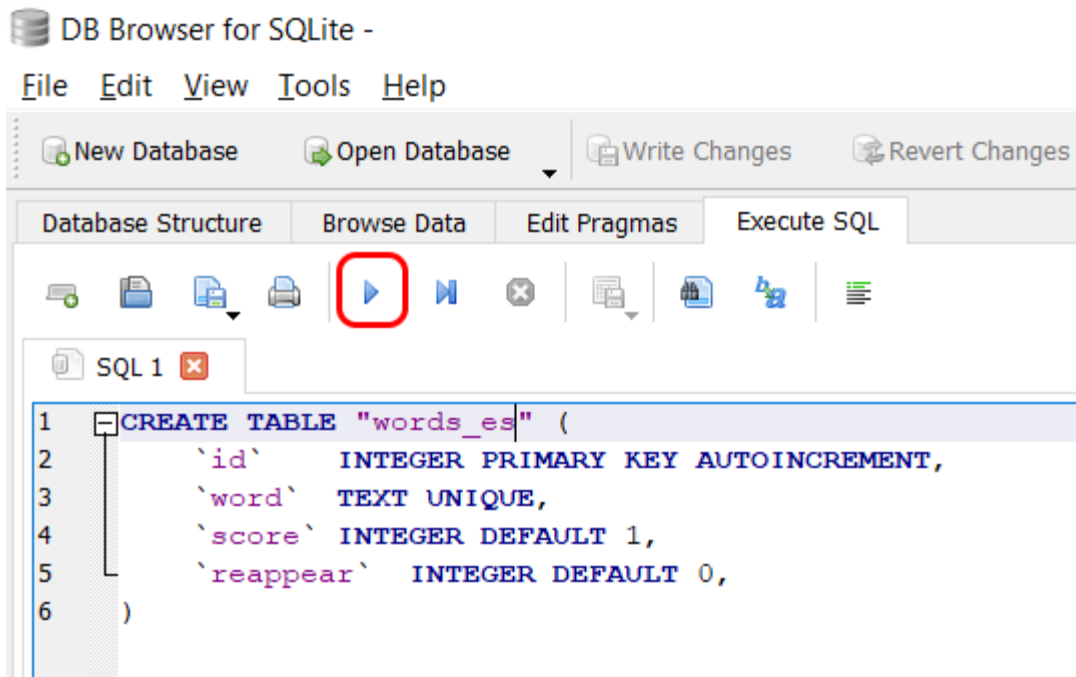


these words? Into the Sqlite database. Go to <https://sqlitebrowser.org/dl/> and download Sqlite database editor. Once installed go to File->Open Database and select the database file called word\_source in the db folder. Words for two languages have been prepared already. To see the tables holding these words select any of them from the Table combo box (number 1):



Let's create a table for your language. Go to the Execute SQL tab (number 2 above). Change the "en" in "words\_en" with the 2-letter of your language code such as "pt" or "es". Then paste it into the script pane and execute it (see the screenshot below).

```
CREATE TABLE "words_en" (
  `id` INTEGER PRIMARY KEY AUTOINCREMENT,
  `word` TEXT UNIQUE,
  `score` INTEGER DEFAULT 1,
  `reappear` INTEGER DEFAULT 0,
)
```



This creates a table for your language. Using the combo box mentioned before, open your table and add words. The id field is incremented automatically, so don't change it. Each of your words must have a score. Score marks the difficulty of a word, whether it is used in the game or whether it is a vulgar word. You don't have to score your words if you don't want to (default value is 1), but it adds quality to your game. Here are the possible values and their explanations:

- 1: Vulgar words. Never used in the game. If a user swipes such a word, a warning will appear saying that it doesn't count towards a bonus word.
- 0: Non-familiar or somewhat awkward words. Never used in the game. They count towards a bonus word if the user swipes it.
- 1: Words that should appear in early levels. They are commonly used.
- 2: Words that appear after 1.
- 3: Words that appear after 2.
- 4: They are difficult to find and not very familiar. For eg. technical jargon.

**Note:** The English language table contains many words that are marked as 0. If you want to add more words, you may consider making these a higher number. If you are a native speaker of English you know it better than me whether it is suitable to use these words or not. But there is a catch! The words which have **NULL** on the meaning field miss a definition in the dictionary service. So, don't use these words. Just use the ones that are completely blank on the meaning side.

You won't manually fill the reappear field. The generator fills it for itself. Now this is important: **if you delete English or Turkish words or any other language that you created, and if you want to regenerate the levels, then you must execute the following SQL code (changing the language code).** It resets the reappear field. The

reappear is used to control when that very specific word will appear again in the game. Simply put, it helps to distribute words evenly across levels:

```
update words_en set reappear = 0;
```

However, filling the database rows one-by-one is quite tedious and will take a long time. Instead, you should automate this process. But how? Search the Internet for a list of Spanish words for example. There are ways to find them, I am sure you can find them. I suggest you to format them as a list in notepad++. Each word must go to its own line and all of them must be in capital letters like so:

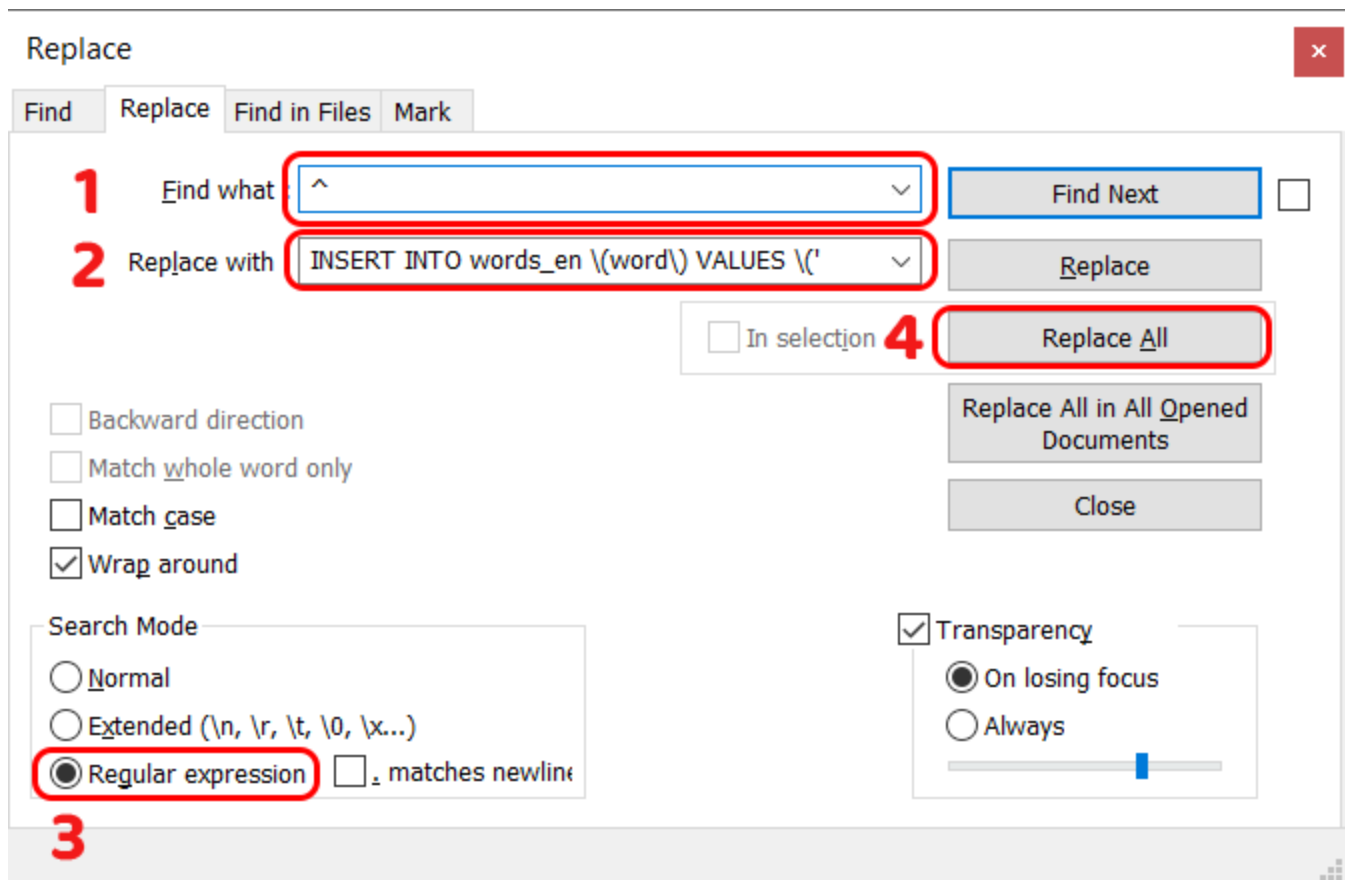
```
MYWORD  
ANOTHERWORD  
THATWORD  
THISWORD  
GOODWORD
```

If you find and make your words in this format, excellent. But note that you may need thousands of them. So far so good. Now we need a way to insert these into the database at once. We will do it using the regular expression magic. Make sure that you are using notepad++ (<https://notepad-plus-plus.org/downloads/>). For Mac OS, please use a text editor that supports regular expression.

Now, all words are open in Notepad++, on your keyboard hit CTRL + H to open the replace dialog. In the Find What box, enter a caret (^ circumflex), on my keyboard I do it by SHIFT + 3 then SPACE. Paste the following into the Replace With field (change the en part with your language code):

```
INSERT INTO words_en \ (word\ ) VALUES \ ('
```

Make sure regular expression is selected and click Replace All.



This will give the following result:

```
INSERT INTO words_en (word) VALUES ('MYWORD
INSERT INTO words_en (word) VALUES ('ANOTHERWORD
INSERT INTO words_en (word) VALUES ('THATWORD
INSERT INTO words_en (word) VALUES ('THISWORD
INSERT INTO words_en (word) VALUES ('GOODWORD
```

^ means the beginning of line, so we replaced it. Time to replace the end part. Enter a dollar sign (\$) in the Find What field. For the Replace With, paste the following:

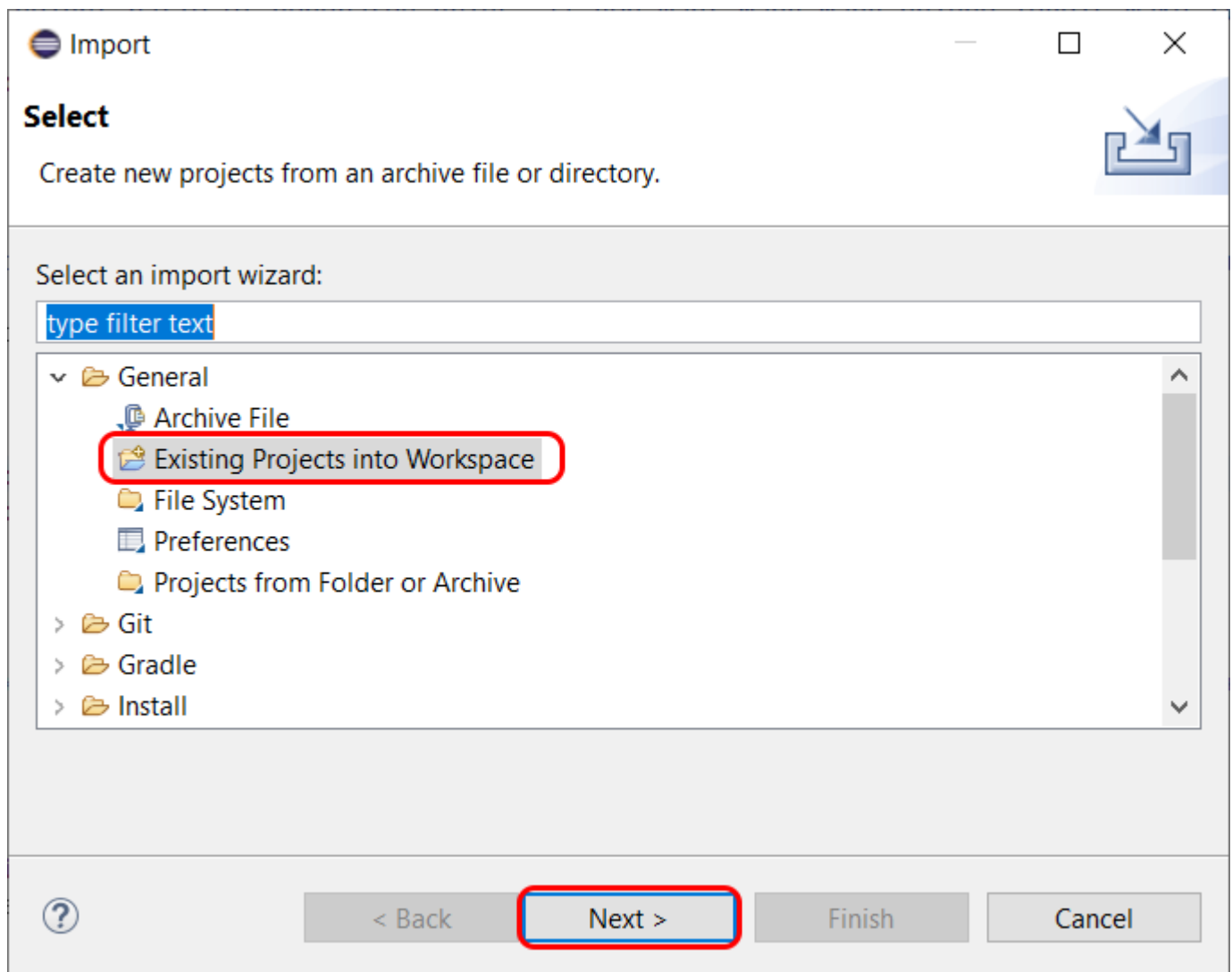
```
'\);
```

And we get:


```
INSERT INTO words_en (word) VALUES ('MYWORD');
INSERT INTO words_en (word) VALUES ('ANOTHERWORD');
INSERT INTO words_en (word) VALUES ('THATWORD');
INSERT INTO words_en (word) VALUES ('THISWORD');
INSERT INTO words_en (word) VALUES ('GOODWORD');
```

Close the replace dialog, copy the resulting text into the SQL pane of SQL editor and run it. These words are now populated in the database. Quite handy. If you find all the words, it is just a few find and replace operations to fill the database.

2. Download the Eclipse IDE at <https://www.eclipse.org/downloads/>.
3. Set it up or unzip it to a suitable location on your PC.
4. Once ready, go to File->Import... This will lead you to the following:





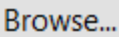
Make sure the option "Existing Projects into Workspace" is selected and click Next. In the next dialog box select the root folder of the Eclipse generator project folder named CrosswordGenerator. This folder is in the zip files you downloaded from Codecanyon. Then click finish:


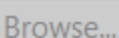
 Import

## Import Projects

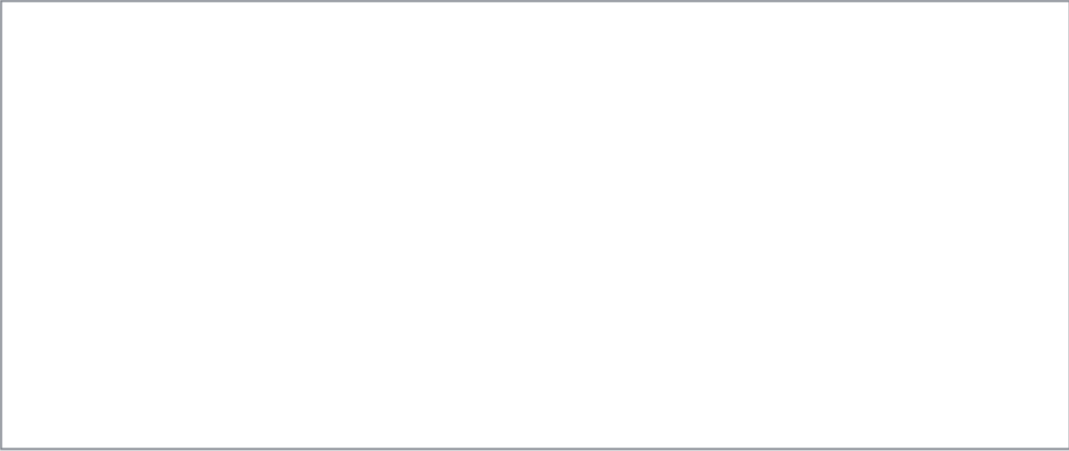
Select a directory to search for existing Eclipse projects.

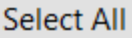
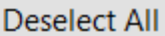



☒ Select root directory:   

☐ Select archive file:   

Projects:






  
  







Options

- ☐ Search for nested projects
- ☒ Copy projects into workspace
- ☐ Close newly imported projects upon completion
- ☐ Hide projects that already exist in the workspace

Working sets

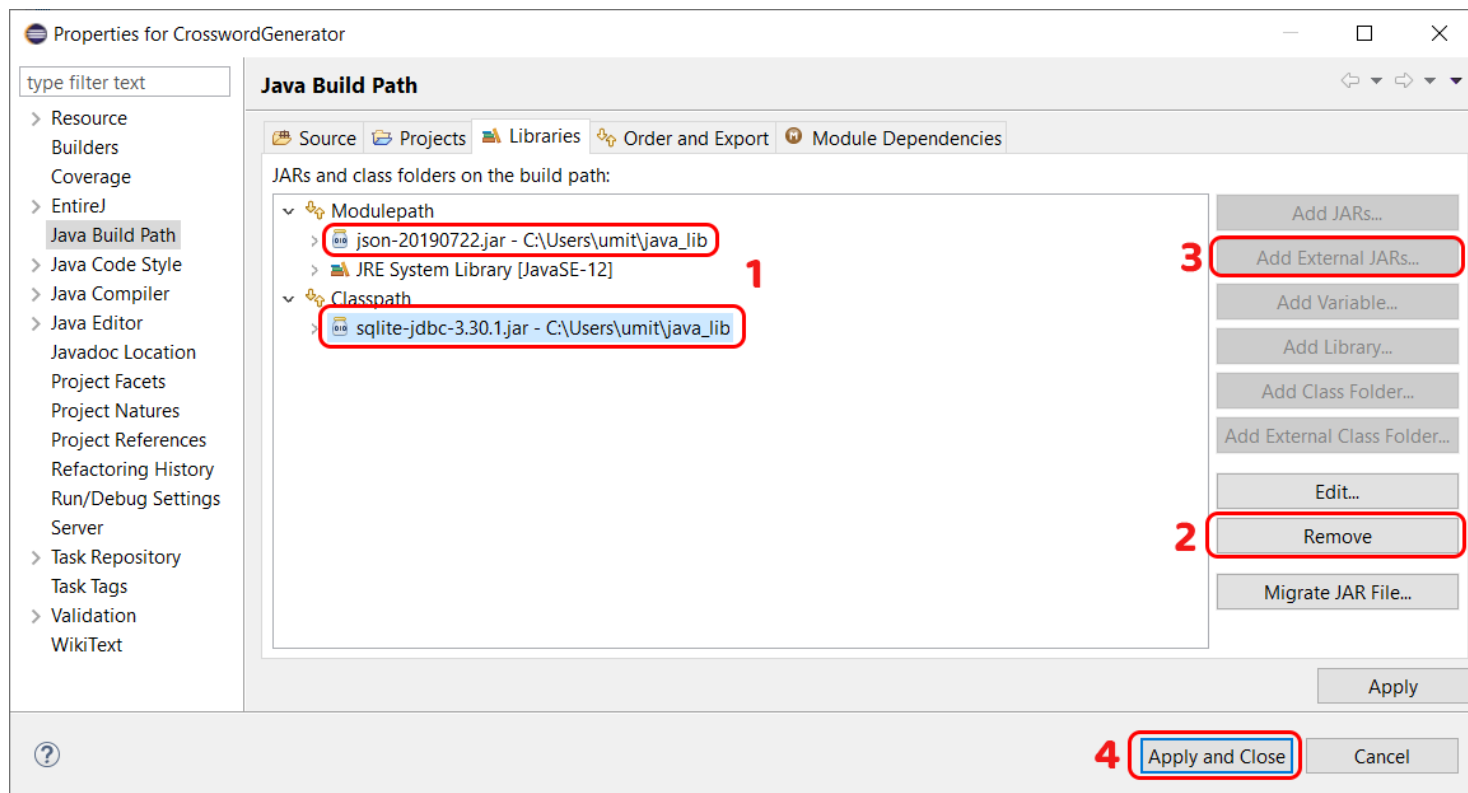
☐ Add project to working sets 

Working sets:   

After this process, the project will be placed on the left part of Eclipse IDE. However, some

errors will pop up because it can't find the dependencies it needs. In the zip file you downloaded from Codecanyon, there is a folder called eclipse\_dependencies, there are two files in this folder, we should make the Eclipse project find these files. In the project pane, right click the top project folder and choose Build Path->Configure Build Path. Such a dialog will open:

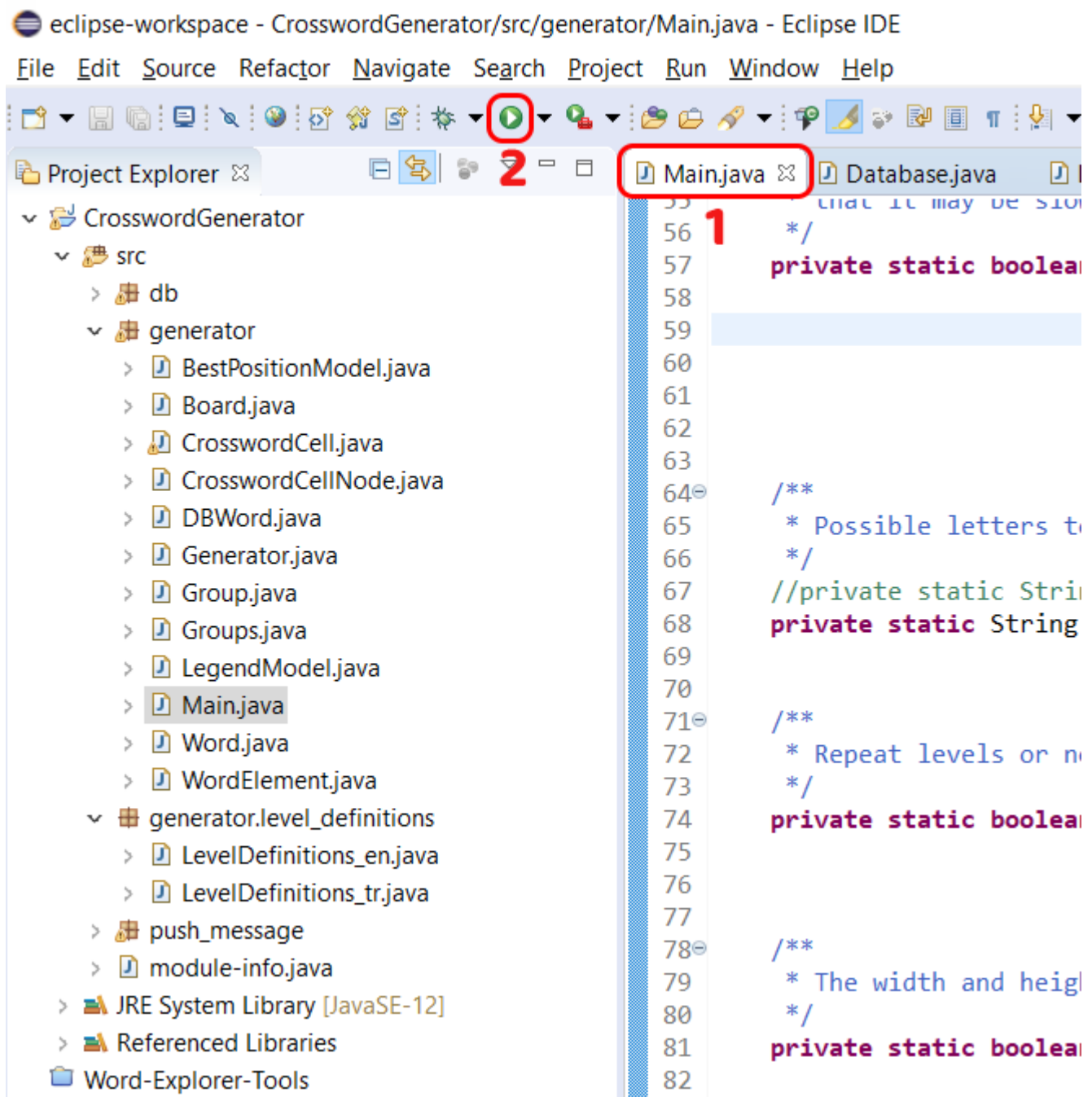


Select the problem files (1) and remove them (2). Then, select the files on your hard drive clicking 3 and finally click Apply and Close (4). The errors should disappear.

In the project pane, open src/generator/Main.java. There are configuration options on this page and they are fully commented. Change them according to your needs. But how shall we customize levels so that some will have 3 letters on the dial, some 7, some boards will have 4 words and some 15? The word connect game theoretically supports customization of each level individually but in practice this is not feasible. Therefore, we need to group similar levels and apply the same configuration for that group. To understand what I mean open the file src/generator/level\_definitions/LevelDefinitions\_en.java. This file describes how levels will be constructed for the English Language. Please examine the comments and the source code below the page. Configuration is set according to level index. Now, you need a file similar to this. In the project pane of Eclipse, right click on this file, copy and paste it into the same folder but change the language code in the name. This file may work for your language or it may not.

Everything should be ready to generate the levels now. The generator tool exports each level in its own file rather than all in one, to save memory at run time. Open

the Main.java file mentioned before and run the program:



You can see information about the generated and failed levels in the console output. If the number, variety and score of your words do not match the level definitions, the generation may stall. In such a case, you should stop the program (by clicking the red button on the console output) and review your your level definitions. **When you finish generating levels, you must inform the game about the total number.** Go to Android Studio,



open `core/src/word.game/config/GameConfig` and find the `availableLanguages` variable. Create a new entry similar to the existing "put" statements with your language code and enter the total number of levels. Be careful: don't enter the number of the final level, total number is final level index + 1.

5. UI Text. It is time to translate the UI labels to your new language. Go to `android/assets/data/en`, copy `strings.properties` and paste it into your new language folder you created in step 1. Open it in Android Studio or another text editor. If you create a brand new file don't forget to make its encoding **UTF-8**, otherwise your non-ascii characters may not be visible on GUI. You shouldn't delete the numerical parameters that take place in curly braces while translating the text. The following table explains where each string takes place:

<code>play_label</code>	Play button in the intro screen
<code>next_level</code>	The button that appears at level end
<code>back</code>	The button that show when all levels are consumed
<code>no_connection</code>	Appears when there is no Internet connection
<code>dictionary</code>	Dictionary title
<code>word_not_found</code>	Appears when a word is missing in dictionary
<code>no_response</code>	Appears when the dictionary server fails to respond
<code>not_extra_word</code>	Appears when the player swipes a vulgar word
<code>extra_words_incomplete</code>	Bonus words dialog title
<code>extra_words_complete</code>	Bonus words reward dialog title
<code>extra_words_collected</code>	Appears in bonus words dialog
<code>extra_words_in_this_level</code>	Appears in bonus words dialog
<code>claim</code>	Claim
<code>lucky_wheel_dialog_title</code>	Lucky wheel title
<code>spin_btn_label</code>	Lucky wheel spin button label
<code>tap_to_collect</code>	Appears when lucky wheel spin ends
<code>spin_again</code>	Lucky wheel spin button label
<code>spin_tomorrow</code>	Appears at lucky wheel dialog after all spins
<code>menu</code>	Game menu dialog title
<code>language</code>	Language dialog title (also menu item)
<code>gdpr</code>	Menu item
<code>sounds</code>	Menu item (mute)

rate_us	Menu item
contact_us	Menu item
finger_hint_msg	Appears on finger hint button click
iap_error	IAP error dialog title
iap_error_text	IAP error dialog text
okay	Alert dialog button
shop	IAP dialog title
coins	Appears on the coins ribbon of IAP
combo_packs	Appears on the bundles ribbon of IAP
one_time_packs	Appears on the one time purchase of IAP
free_hints	Watch and earn coins dialog title
ad_invite	Watch and earn coins dialog text
watch	Watch and earn coins dialog button label
no_video	Appears when rewarded video ad is not ready
remove_ads	Remove Ads dialog title
remove_ads_desc	Remove Ads dialog text
buy	Remove Ads dialog button label
confirm	Confirm dialog title
home_return	Appears on confirm dialog when back the button of android is clicked
yes	Confirm dialog button label
no	Confirm dialog button label
rewarded_ad_amount	Appears after watching a reward ad to the end
rate_us_title	Rate us dialog title
rate_us_text	Rate us dialog text
rate_button_label	Rate us dialog button label
later_button_label	Rate us dialog button label
combo_side	Appears on the left side of the screen when the user makes a combo
combo_top	Appears at the top of game screen to specify combo count
combo_earned	Appears at the end of a level to specify how many coins were earned for collecting combo.
feedback	Appears on a ribbon after each combo the user achieves
level	Appears at the top of game screen to specify the current

	level number
how_to_play	How to play dialog title
how_to_play_desc1	How to play dialog text
how_to_play_desc2	How to play dialog text
how_to_play_desc3	How to play dialog text
bomb_exploding	Bomb dialog title
out_of_moves	Bomb dialog text
watch_video	Bomb dialog button label
bomb_defused	Appears on ribbon when the bomb is disarmed
failed_bomb	Bomb blasted dialog title
blasted	Bomb blasted dialog text
retry_bomb	Bomb blasted dialog button label
app_version	Appears at the bottom of game menu
hint_locked	Appears on tooltip when it is too early to use a hint
dial_tutorial_1	Used for the first word of the dial tutorial
dial_tutorial_2	Used for the second word of the dial tutorial
shuffle_tutorial	Shuffle button tutorial text
single_random_hint_tutorial	Single random hint button tutorial text
finger_hint_tutorial	Finger hint button tutorial text
multi_random_hint_tutorial	Multi random hint button tutorial text
rocket_hint_tutorial	Rocket hint button tutorial text
got_it	Got it button label of some tutorials
ufo_tutorial	UFO reward tutorial text
bomb_tutorial	Bomb tutorial text
goldpack_tutorial	Gold pack reward tutorial text
monster_tutorial	Monster reward tutorial
bonus_word_tutorial_1	Appears on the first discovery of a bonus word
bonus_word_tutorial_2	Appears when the bonus words dialog is opened for the first time
to_be_continued	Appears at the end of the final level in the game
not_available_for_rocket	Appears when there is not a suitable word to deliver rocket (all words are filled of are hinted or a booster on them)

6. Create some graphics for the new language. You need an icon for the language menu.

It has already been explained how to add a new graphic file. Go to `sprite_sheets/atlas_4` folder. In that folder, you will see a Turkish and a US flag. You need one with the flag of your language. Make it the same shape and size. You will also create a LEVEL CLEAR ribbon text in photoshop (See the `cup.psd` file for this). Don't forget to name these files similar to the existing ones. Now, include these 2 files in the texture packer and export the updated sprite sheet. You don't need to do anything in code for these two graphics. The system will recognize them from the the language code part of their file names (you entered the new language into the `availableLanguages` variable, that's why).

7. The dictionary. When players find a word on the crossword grid and tap on them, a dictionary dialog opens with a definition of that word. There may be very-little used, obscure words and when players don't know the meaning, they wish to learn it. Therefore, most word games offer such a functionality. Our word connect game has a dictionary for the English language only. For Turkish, there is only one dictionary available (TDK) but it wasn't used due to copyright restrictions.

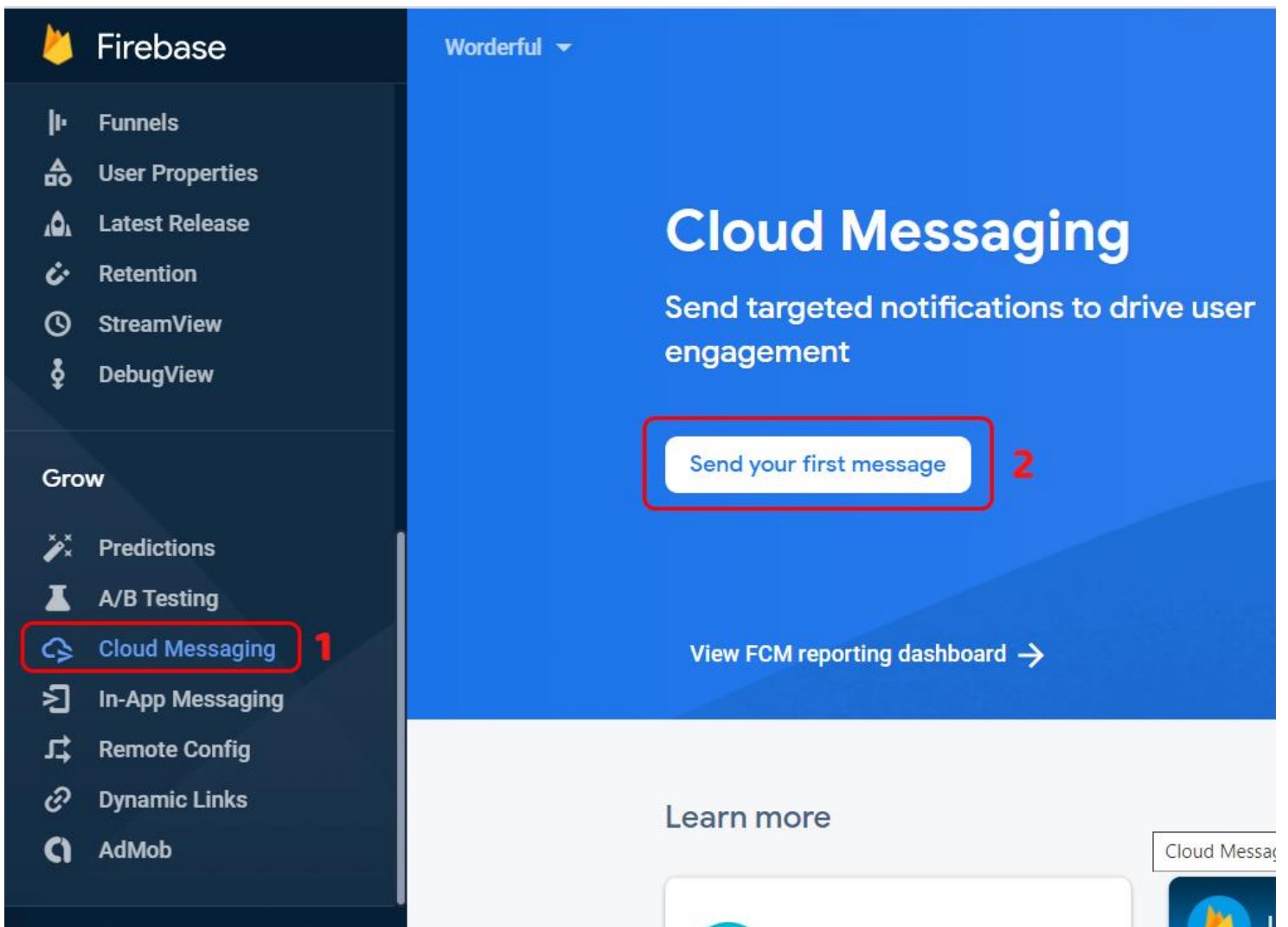
You must have a web service running on a server that will respond with a definition when the meaning of a word is asked. You may run your own server or get a paid service. It is unlikely for most customers of this product to have such an availability. But what if you have it? You need some programming knowledge to send an HTTP request and forward the response to the dictionary dialog. In Android Studio open the file `android/src/word/game/WordMeaningRequest_en`. This file handles the HTTP requests for the English language, examine it. The functionality in this file uses the Libgdx network API. It is simple and convenient. The important function is `parseResponse`. Various dictionary servers may respond the definition in different formats such as HTML, JSON or XML. Here you the server response is converted to the format that our dictionary dialog understands. The dictionary used for English (wordnet) responds with an HTML page. To convert this such a response for our dialog, I used the Java Jsoup library (<https://jsoup.org/>). This library parses, strips out unnecessary parts of an HTTP web page according to our needs. When parsing is done we notify the dictionary dialog about this word and its definition so that it can display them. We do this using the callback function named `onMeaning`. It takes 2 parameters: 1-The word, 2-The definition acquired for the word. If you want to continue, save as the `WordMeaningRequest_en` in the same folder, changing the file name. Implement your request and response mechanism. Now, open the `android/src/word/game/WordMeaningProviderAndroid` source file. You need to enter your new request file into the `get` function in this file. Imagining you implemented a request file for German, it should look like this:

```
public WordMeaningRequest get(String langCode){
    if(langCode.equals("en")) return new WordMeaningRequest_en();
    if(langCode.equals("de")) return new WordMeaningRequest_de();
    return null;
}
```

## **SENDING PUSH MESSAGES**

You can send push messages to all of your users easily, awarding them with free coins. Some app in Play Store do this on special days. There are some steps to fulfill for this.

1. Go to the Firebase console and create a new project for your game using your package id you created before.
2. When create the project, Firebase will create a json file named google-services.json. Download this json file and copy it to the android folder and overwrite the existing one. The users of your app must open the app at least once for this to work. Opening the app subscribes the device for your push messages.
3. You are now ready to send push messages to your app. Your app can receive the messages in 2 ways. 1-When the app is closed. When the app is closed, the device will receive the PM into its status bar and when clicked it will open the game adding the coins you sent. 2-When the app is running. The player is informed with an informative dialog box about this while he/she is playing the game.
4. To send a PM go to the Firebase console. When the project is selected, go to the Grow menu on the left and click on Cloud Messaging, then click on Send your first message:



Fill in the form that opens. In the first section you will fill in the Notification title and the Notification text fields. Go to the second section named Target. Select the Topic option and paste "coin\_topic" into the Message topic field. Now skip to the fifth section. For the Key section of the Custom data paste "coins". For the Value field type the number of coins you want to send to users, say, 500. Enable the Sound field so that it will draw the attention of the player. You are now ready to send the message. Click the Review button on the bottom-right part of the form and then click the Publish button on the opening dialog box. You can copy and use this as a template for your subsequent push messages, using the duplicate option you will see. You should test this functionality on your self before publishing the app to the Play Store.

## DEBUGGING THE GAME

When something is not working you should try to find the cause using the Logcat output of Android Studio. When your game is running, open the Logcat by View->Tool Windows->Logcat:



1- Select the emulator or the device that is running the game. 2- Select the package id of your game. 3- Debug should be selected. 4- It depends. 5- It should be selected. Number 4 depends on the thing that is not working. The following table lists the possible things you can enter into this field:

interstitial_ad	Related to interstitial ads
rewarded_ad	Related to rewarded ads
gdpr	Related to GDPR
fcm	Related to Firebase messaging
iap	Related to In-app purchases
game.log	Output the missing levels and level answers

When you don't have a problem with the app or you can't find the cause of the problem with these, it is best to leave the field number 4 empty, so that it will output all debug messages.

### COPYRIGHT INFORMATION

Product	License
Libgdx game engine	Apache 2
Jsoup	MIT
JDBC Driver	Apache License
JSON	Json License
Crossword generator <a href="https://github.com/satchamo/Crossword-Generator">https://github.com/satchamo/Crossword-Generator</a>	MIT
Princeton University Wordnet Dictionary <a href="https://wordnet.princeton.edu/license-and-">https://wordnet.princeton.edu/license-and-</a>	WordNet 3.0 license

commercial-use	
Montserrat fonts	Google open source license
Language icon <a href="https://www.iconfinder.com/icons/897244/language_courses_learn_speak_icon">https://www.iconfinder.com/icons/897244/language_courses_learn_speak_icon</a>	CC BY 3.0
Sound icon <a href="https://www.iconfinder.com/icons/4105545/melody_sound_audio_music_icon">https://www.iconfinder.com/icons/4105545/melody_sound_audio_music_icon</a>	Free for commercial use
Rate us icon <a href="https://www.iconfinder.com/icons/2135937/pluse_favorite_game_star_rate_icon">https://www.iconfinder.com/icons/2135937/pluse_favorite_game_star_rate_icon</a>	CC BY 3.0
Email icon <a href="https://www.iconfinder.com/icons/3213310/mail_message_email_letter_icon">https://www.iconfinder.com/icons/3213310/mail_message_email_letter_icon</a>	CC BY 3.0
Dictionary icon <a href="https://www.flaticon.com/free-icon/dictionary_1680931">https://www.flaticon.com/free-icon/dictionary_1680931</a>	Free for personal and commercial use
Coins <a href="https://opengameart.org/content/coin-animation">https://opengameart.org/content/coin-animation</a>	Free for commercial use
Gift box <a href="https://pngtree.com/freepng/chest-open-and-close-with-gold-coins_5056779.html">https://pngtree.com/freepng/chest-open-and-close-with-gold-coins_5056779.html</a>	Free license (with attribution)
IAP ribbon <a href="https://pngtree.com/freepng/blue-ribbon-border_3091663.html">https://pngtree.com/freepng/blue-ribbon-border_3091663.html</a>	Free license (with attribution)
Coins <a href="https://pngtree.com/freepng/a-bunch-of-coins_4087919.html">https://pngtree.com/freepng/a-bunch-of-coins_4087919.html</a>	Free license (with attribution)
Chest <a href="https://pngtree.com/freepng/financial-treasure-chest-decoration-illustration_4580876.html">https://pngtree.com/freepng/financial-treasure-chest-decoration-illustration_4580876.html</a>	Free license (with attribution)
UFO <a href="https://bevouliin.com/flying-heroes-spaceship-attack/">https://bevouliin.com/flying-heroes-spaceship-attack/</a>	Free for personal and commercial use
Bomb <a href="https://pngtree.com/freepng/cartoon-bomb-background-and-explosive-light-vector---illustration_4977135.html">https://pngtree.com/freepng/cartoon-bomb-background-and-explosive-light-vector---illustration_4977135.html</a>	Free license (with attribution)



Settings icon <a href="https://www.flaticon.com/free-icon/settings_148912?term=settings&amp;page=1&amp;position=19">https://www.flaticon.com/free-icon/settings_148912?term=settings&amp;page=1&amp;position=19</a>	Free for personal and commercial use (with attribution)
Close icon <a href="http://www.clker.com/clipart-241419.html">http://www.clker.com/clipart-241419.html</a>	Free for anything
Bolt <a href="https://pngtree.com/freepng/thunder-and-bolt-flash-vector_4723242.html">https://pngtree.com/freepng/thunder-and-bolt-flash-vector_4723242.html</a>	Free license (with attribution)
Cup <a href="https://pngtree.com/freepng/gold-trophy-with-the-name-plate-of-the-winner-of-the-competition_5299953.html">https://pngtree.com/freepng/gold-trophy-with-the-name-plate-of-the-winner-of-the-competition_5299953.html</a>	Free license (with attribution)
Padlock <a href="https://pngtree.com/freepng/golden-lock-icon-vector_5066545.html">https://pngtree.com/freepng/golden-lock-icon-vector_5066545.html</a>	Free license (with attribution)
Arrow <a href="https://pngtree.com/freepng/green-game-arrow_5409447.html">https://pngtree.com/freepng/green-game-arrow_5409447.html</a>	Free license (with attribution)
Lucky Wheel <a href="https://pngtree.com/freepng/golden-spin-wheel_4199603.html">https://pngtree.com/freepng/golden-spin-wheel_4199603.html</a>	Free license (with attribution)
Background images <a href="https://www.pexels.com/license/">https://www.pexels.com/license/</a>	Free license
Thanksgiving background <a href="https://www.freepik.com/free-psd/frame-with-thanksgiving-day-message_5657290.htm#page=2&amp;query=thanksgiving&amp;position=24">https://www.freepik.com/free-psd/frame-with-thanksgiving-day-message_5657290.htm#page=2&amp;query=thanksgiving&amp;position=24</a>	Free for personal and commercial use (with attribution)
Beach background <a href="https://www.freepik.com/free-photo/top-view-sand-meeting-seawater_9819012.htm#page=1&amp;query=sand%20beach&amp;position=48">https://www.freepik.com/free-photo/top-view-sand-meeting-seawater_9819012.htm#page=1&amp;query=sand%20beach&amp;position=48</a>	Free for personal and commercial use (with attribution)
Blast sound fx <a href="https://freesound.org/people/eardeer/sounds/402006/">https://freesound.org/people/eardeer/sounds/402006/</a>	CC BY 3.0
Hint sound fx <a href="https://freesound.org/people/GabrielAraujo/sounds/242501/">https://freesound.org/people/GabrielAraujo/sounds/242501/</a>	CC0 public license
Level end sound fx	CC BY 3.0

<a href="https://freesound.org/people/elijahdanie/sounds/487436/">https://freesound.org/people/elijahdanie/sounds/487436/</a>	
Lucky wheel click sound fx <a href="https://freesound.org/people/Agaxly/sounds/217482/">https://freesound.org/people/Agaxly/sounds/217482/</a>	CC0 public license
Shuffle sound fx <a href="https://freesound.org/people/g_e_n_e/sounds/485010/">https://freesound.org/people/g_e_n_e/sounds/485010/</a>	CC BY 3.0
Notification sound fx <a href="https://freesound.org/people/YourFriendJesse/sounds/235911/">https://freesound.org/people/YourFriendJesse/sounds/235911/</a>	CC0 public license