МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ “ВОЛОЧАЕВСКИЙ ЛИЦЕЙ”

ИНДИВИДУАЛЬНЫЙ ИТОГОВЫЙ ПРОЕКТ

**Тема:** “Создание игры с помощью Java фреймворка libGDX”

Выполнил обучающийся:  
11.2 класса  
Решетников Сергей Валерьевич

Научный руководитель:  
учитель информатики  
Людмила Викторовна  
Дементьева

г. Хабаровск  
2021-2022 учебный год.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc87036545)

[Глава 1. Настройка проекта 4](#_Toc87036546)

[Изучение Java 4](#_Toc87036547)

[IDE 4](#_Toc87036548)

[Установка Java (рекомендуется) 5](#_Toc87036549)

[Генерация проекта 6](#_Toc87036550)

[Импорт проекта 8](#_Toc87036551)

[Запуск проекта 9](#_Toc87036552)

[Что делать, если не удалось запустить проект? 9](#_Toc87036553)

[Итог 1 главы 9](#_Toc87036554)

[Глава 2. Создание базовой версии игры. 11](#_Toc87036555)

[Жизненный цикл игры 11](#_Toc87036556)

# ВВЕДЕНИЕ

Данная работа посвящена изучению Java, а конкретнее одного из ее фреймворков под названием “libGDX”. **Тема** данного проекта это и отражает: “Создание игры с помощью Java фреймворка libGDX”.  
Собственно, данный фреймворк и язык являются как и **объектами** исследования, так и **предметами** исследования.  
**Актуальность** данного проекта заключается в том, чтобы научить кого-либо прототипировать и создавать базовые приложения на этом фреймворке т.к он позволяет создавать не только игры, а также различные приложения, которые могут значительно облегчить нашу жизнь, а самый легкий способ чему-то научиться, это синхронно повторять действия за кем-то, параллельно понимая для чего они делаются.  
**Проблема** данного проекта заключается в нехватке навыков у некоторых людей для создания чего-либо с помощью Java, что данный проект и пытается решить.  
**Цель** проекта: научиться прототипировать приложения на libGDX, а также параллельно создать версию игры “Uno”.  
**Задачи** этого проекта:

* Установить необходимые программы для прототипирования.
* Рассмотреть базовые вещи в IDE (todo:)
* Разработать первую игру на libGDX.

На случай путаницы, был создан специальный Git репозиторий на платформе GitHub, в котором по коммитам можно рассмотреть мое неотсортированное создание приложения и документации по шагам.

Гипотеза: зачем она нужна?

Практическая значимость исследования?

Этот проект собой подразумевает то, что вы абсолютный новичок в программировании на Java и libGDX. Если же Вы знаете основные методологии разработки на Android, Вам хватит и документации.

[Гайд на русском](http://www.libgdx.ru/p/guide.html) | [документация на английском](https://github.com/libgdx/libgdx/wiki/).

# Глава 1. Настройка проекта

## Изучение Java

Несмотря на то, что этот проект был создан для абсолютных новичков, Вам все же необходимы знать базовые элементы языка Java. Я не смогу тут рассмотреть изучение Java как языка, но могу порекомендовать [платформу Metanit](https://metanit.com/java/tutorial/) для изучения абсолютных основ данного языка, т.к иначе Вы рискуете ничего не понять. Из IDE советую использовать **Intellij Idea.**

## IDE

Для начала, вам нужно установить **IDE** (**Integrated Development Environment**). Это что-то вроде редактора для всех ваших файлов, а также контейнера для нужного софта вроде компилятора, Gradle и т.п..

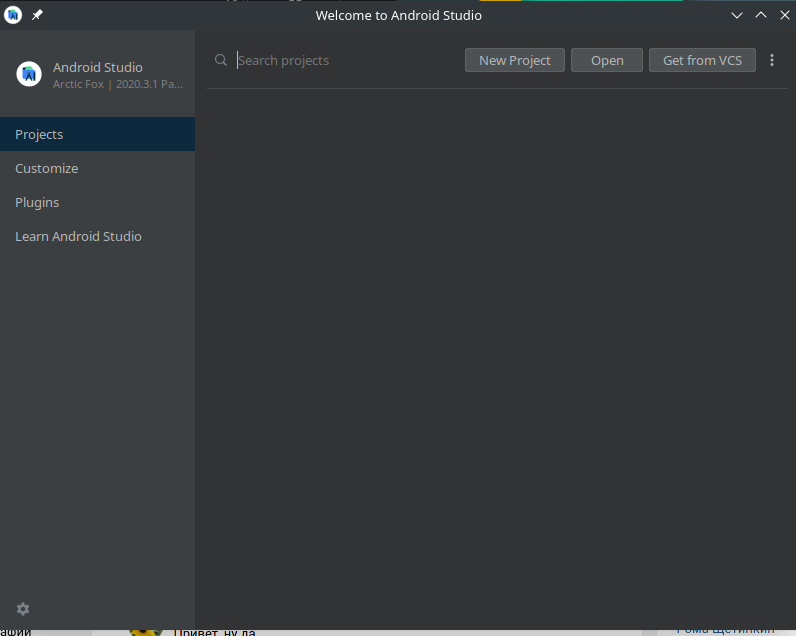
Если IDE у Вас уже установлена, Вы можете смело пропустить [эту часть.](#Java)

Есть множество вариантов, вроде **Android Studio**, **Intellij Idea**, **Eclipse**, или даже можно воспользоваться обычным блокнотом.

Но я Вам рекомендую установить [**Android Studio**](https://developer.android.com/studio),так как:

* Мы будем компилировать приложение на Android, а отлаживать приложение с него удобнее всего.
* Все примеры в данном проекте будут основаны на нём, так что следовать инструкциям будет проще.
* Она бесплатна, и максимально проста для новичков.

После установки, запуска и стартовой настройки программы, у Вас высветится окошко с проектами наподобие этого:



Смело закрывайте окно. Вместо использования **Android Studio** для генерации проекта, мы будем использовать специальную утилиту от Libgdx, так как она упрощает создание проекта.

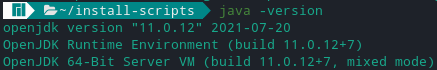
## Установка Java (рекомендуется)

Несмотря на предустановленную JDK (Java Development Kit) в Android Studio, я Вам также рекомендую установить Java 15 т.к она обладает новыми функциями, которые мы собираемся использовать. Вы можете также создать проект с Java 8-14, но тогда могут возникнуть ошибки в синтаксисе, а также проблемы с оптимизацией при точном следовании инструкции. (Java 16 и выше, по какой-то причине не поддерживаются libGDX),

Я Вам рекомендую пользоваться [AdoptOpenJdk](https://adoptopenjdk.net/releases.html), или же [liberica-jdk](https://libericajdk.ru/pages/downloads/) от Bellsoft.

После установки рекомендую проверить открыть командную строку/консоль, и проверить ее версию. (в windows: Win + R – cmd.exe открывает командную строку)

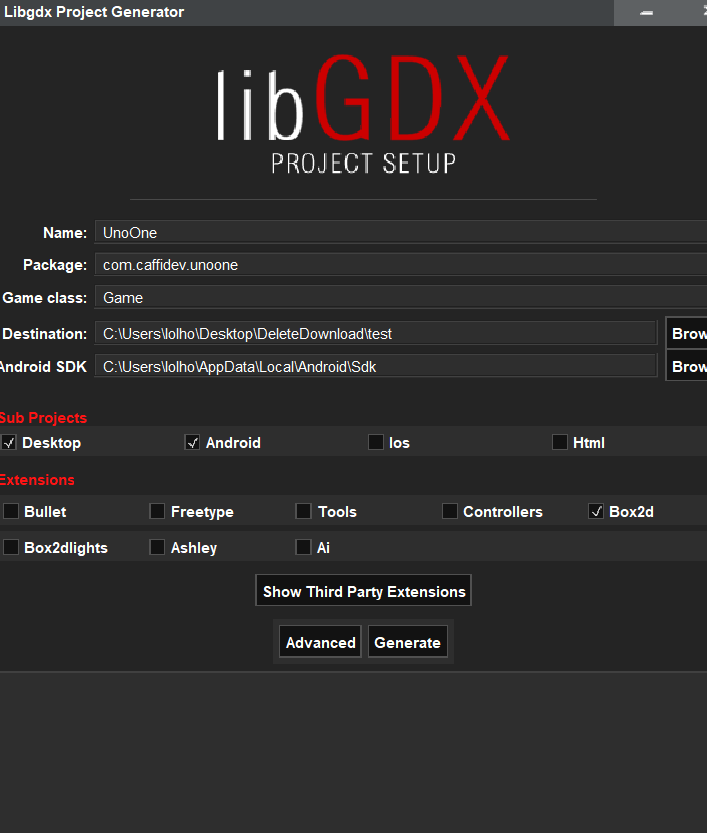
Проверить версию можно командой: java -version

  
*у меня jdk 11 как основная, не бойтесь если показана другая версия, вы в IDE всегда можете ее сменить*

Если вылезла ошибка о том, что такой команды нет, то либо Вы неверно ее установили, либо у Вас не прописана она в [переменной среды](https://java-lessons.ru/common-errors/javac-is-not-recognized).

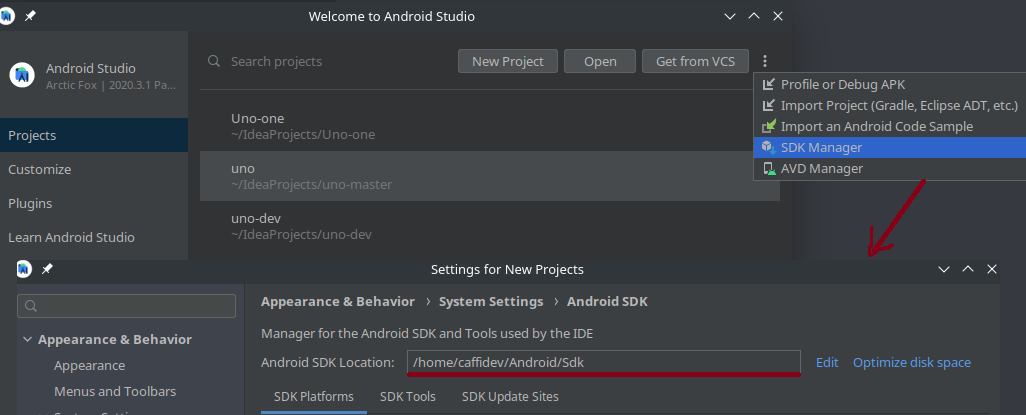
## Генерация проекта

Дальше Вам надо скачать специальную [утилиту](https://libgdx.com/assets/downloads/legacy_setup/gdx-setup_latest.jar) для генерации проектов libGDX.  
После ее скачивания, откройте консоль в папке, куда вы поместили файл и введите команду `java -jar [имя вашего файла]`.



Должен открыться следующий GUI (графический интерфейс)

У Вас спрашивают следующие параметры:

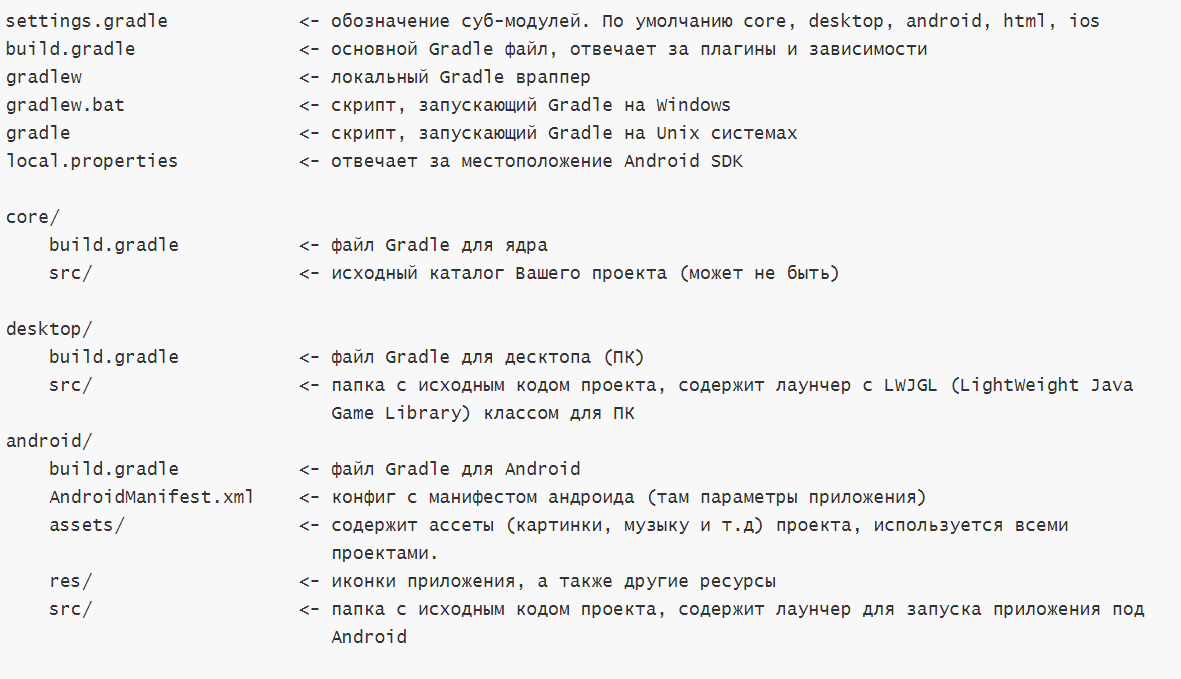
* Name: название Вашего приложения (лучше использовать kebab-case) (unoone)
* Package: Java пакет, в котором будет находится ваш код (com.{username}.unoone)
* Game Class: название основного класса (Game)
* Destination: папка, где код Вашего приложения будет создан.
* Android SDK: местоположение Android SDK.  
  Вы можете его найти его так:

Sub Projects: libGDX кроссплатформен и сам по себе построен на C/C++, а Java – скриптовый язык в нем. Тут мы можем настроить платформы, на которые мы будем компилировать код. Я выбрал Android и Desktop, но по желанию Вы можете выбрать и другие.

Extensions: расширения. В этом гайде они не понадобятся, за исключением Box2D. Подробнее Вы о них можете узнать тут в [документации](https://github.com/libgdx/libgdx/wiki/)[[1]](#footnote-1).

Чтобы сгенерировать проект, нажмите кнопку Generate.

Это сгенерирует директорию с данной расстановкой:



## Импорт проекта

1. Откройте файл build.gradle с помощью Android Studio в директории, в которой Вы сгенерировали проект.
2. Есть вероятность, что некоторые зависимости еще не скачались, поэтому возможно Вам придется перезапустить проект.  
   Чтобы это сделать, нажмите кнопку “Reimport all Gradle projects” в вкладке Gradle, которую можно открыть с помощью “View” -> “Tool Windows” -> Gradle.
3. В этой же вкладке, с версии Android Studio 4.2, задачи Gradle по умолчанию больше не видны. Чтобы их включить, зайдите в “Settings” -> “Experimental” и отключите галочку “Do not build Gradle task during Gradle sync”, после чего синхронизируйте проект с помощью “File” -> “Sync Project with Gradle Files”.
4. Я Вам также советую обновить плагин “Android Gradle Plugin” для наилучшей работоспособности.
5. На Android 12, при компиляции вылезает ошибка в манифесте, в которой сказано что в манифест нужно добавить параметр ‘android:exported’.
   1. Чтобы это сделать, зайдите в манифест AndroidManifest.xml
   2. Добавьте после android:name=”” android:exported=”true”
   3. Этот параметр отвечает за доступность компонента приложения другим приложениям.
6. Смените версию Java, если это необходимо todo:
7. Я Вам также рекомендую найти телефон, и подключить его к компьютеру, включив режим отладки в настройках разработчика. Это позволит запускать версию приложения под Андроид нативно, не прибегая к эмулятору.

## Запуск проекта

Проект можно запускать несполькими способами:

1. Меню Gradle:
   1. Откройте вкладку Gradle сбоку от Вашего окна.
   2. Расширьте задачи Вашего проекта, и выберите: desktop/android -> other -> run.
2. Консольная команда:
   1. Откройте Terminal снизу (если его нет, Вы можете добавить его с помощью View).
   2. Введите команду: ./gradlew desktop:run/android:run
3. Меню конфигурации:
   1. Кликните по DesktopLauncher правой кнопкой мыши.
   2. Выберите “Run DesktopLauncher.main()”;
   3. Откройте меню конфигурации сверху.
   4. Убедитесь что в Working Directory стоит ссылка на папку android/assets.
   5. Аналогично можно сделать и с AndroidLauncher, благодаря чему меняя конфигурацию можно будет менять проект двумя нажатиями мыши.

### Что делать, если не удалось запустить проект?

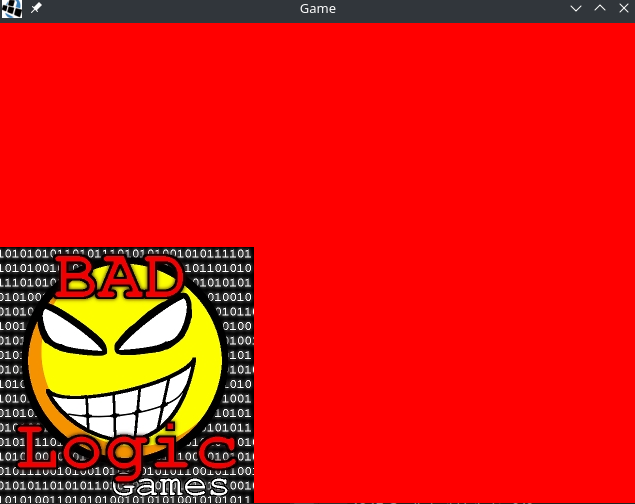
Если ошибка в коде, то исправьте ее по сообщению компилятора, но нежели если ошибка вылезает при билде Gradle, то тогда запустите команду:

./gradlew {платформа}:run , добавив параметр –debug, т.е ./gradlew {плафторма}:run –debug.

Это при повторной компиляции выдаст Вам более подробную ошибку, дав Вам больше информации для решения проблемы.

## Итог 1 главы

Если у Вас все в итоге получилось, Вы должны были увидеть вот эту счастливую мордашку:



Todo: что будет в следующей главе

# Глава 2. Создание базовой версии игры.

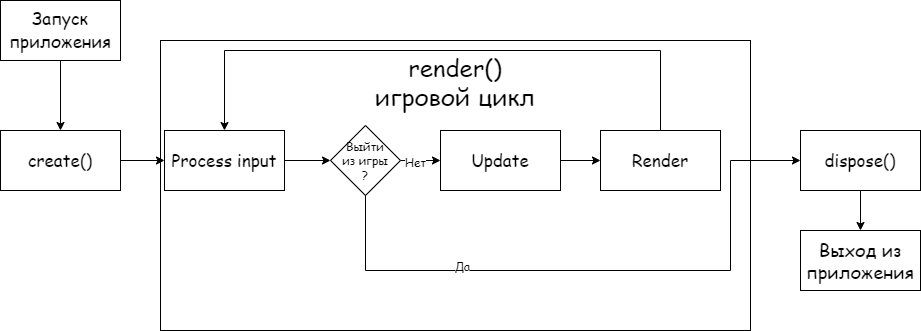
## Жизненный цикл игры

У каждой игры есть свой жизненный цикл. Структура этого цикла зависит от движка, и разработчики должны знать как он проходит. Разумеется, он есть и у libGDX:

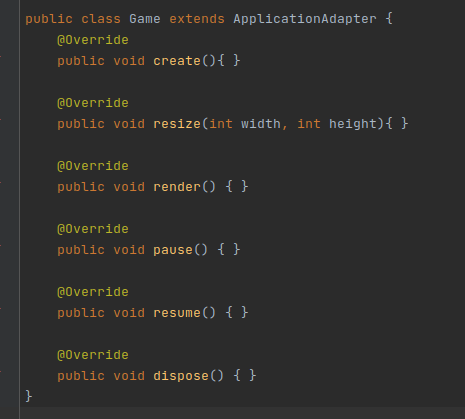
В новой версии libGDX наш основной класс в директории core/ расширяет класс ApplicationAdapter, который является стартовой точкой нашего приложения. Он отвечает за инициализацию нашего приложения, обновление игрового процесса, отображение изображений, установка паузы игры, сохранение состояния приложения и выгрузку ресурсов при выходе из приложения.



Для реализации приложения мы реализуем методы в приложении, которые работают согласно этому жизненному циклу:



Все эти методы содержатся в классе ApplicationAdapter, которые с помощью @override мы можем переопределить, зайдя в наш главный класс под в директории core/ и реализовав данные функции:



Благодаря этим методам мы можем реализовать структуру нашего приложения согласно данной схеме:

* create(): вызывается однократно после запуска приложения, инициализирует нужные классы, переменные, объекты для игрового процесса.
* render() - вызывается каждый раз, когда происходит отрисовка экрана. Вызывается непрерывно, пока мы не решим, что игра закончена. В показанной выше схеме видим, что функция универсальна, и то, что помимо отрисовки экрана мы можем в нее засунуть ввод данных и обновление переменных.  
  На самом деле, разработчик может включить в нее что угодно: музыку, логгинг, получение данных с сервера, и что вообще ему нужно. В этом libGDX и удобен, что в отличие от Unity в нем можно полностью продумать каркас своей программы, несмотря на то, что это дольше и сложнее.
* dispose() - вызывается при закрытии приложения (после метода pause()). Здесь можно подчистить за собой мусор, который не очистил за вас GC (Garbage Collector – сборщик мусора).

Но за что же отвечают остальные элементы?

На самом деле жизненный цикл не настолько прост, т.к помимо самого приложения у нас есть внешняя среда – операционная система, которая им управляет. Если на ПК это Windows/Linux/macOS, где наша программа является процессом, который можно закрыть/открыть/расширить и т.п, то  
в Android вместо процессов используются activity, которые работают по вот этой схеме:



Для таких случаев и нужны следующие методы:

* resize(int width, int height)- вызывается каждый раз, когда приложение меняет размер и оно не находится в состоянии паузы. Так же данный метод вызывается однажды, сразу после вызова метода create(). Передаваемые параметры ширины и высоты меняются в зависимости от размера окна так же каждый раз при вызове метода.
* pause() - вызывается прежде чем приложение будет закрыто. В Android это происходит, когда мы сворачиваем приложение. На пк метод вызывается перед вызовом метода dispose() при закрытии приложения. Обычно в этом методе сохраняют состояние приложение, потому что не факт, что к нему ещё вернуться.
* resume() - вызывается при получении фокуса приложением.

Основные ссылки:

Глава 1:

<https://libgdx.com/dev/setup> - подробная инструкция по установке libgdx (на английском)

<https://github.com/caffidevjava/uno-school> - GIT репозиторий с файлами

<https://metanit.com/java/tutorial/> - Подробное изучение Java  
<https://developer.android.com/studio> - Android Studio.

<https://adoptopenjdk.net/releases.html> - AdoptOpenJdk от Adoptium

<https://libericajdk.ru/pages/downloads/> - Liberica JDK от Bellsoft

<https://java-lessons.ru/common-errors/javac-is-not-recognized> - переменная среды  
<http://www.libgdx.ru/p/guide.html> - Гайд по Libgdx на русском  
<https://github.com/libgdx/libgdx/wiki/> - Документация по Libgdx на английском  
<https://suvitruf.ru/2012/10/20/2274/libgdx-chast-1-zhiznennyj-cikl-igry/> - немного отсюда

1. <https://github.com/libgdx/libgdx/wiki/> [↑](#footnote-ref-1)