МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

**Факультет информационных технологий и робототехники**

Кафедра программного обеспечения информационных систем

и технологий

**Отчет по лабораторным работам**

по дисциплине: “Конструирование ПО”

на тему: ***“ Разработка приложений на языке Java”***

Выполнилстуденты группы *10701421* Крюков А.Ю

Солтан М.В

Принял**:** ст. пр. Станкевич С.Н.

Минск 2023

**Оглавление**

[Лабораторная работа № 01. МЕТОДОЛОГИЯ РАЗРАБОТКИ TEST DRIVEN DEVELOPMENT 4](#_Toc149114771)

[Лабораторная работа № 02. БИБЛИОТЕКИ ПОЛЬЗОВАТЕЛЬСКИЕ 6](#_Toc149114772)

[Лабораторная работа № 03. ПОТОКИ ВВОДА-ВЫВОДА 9](#_Toc149114773)

[Лабораторная работа № 04. ПОТОКИ ВВОДА-ВЫВОДА 11](#_Toc149114774)

[Лабораторная работа № 05. ЖУРНАЛИРОВАНИЕ. 13](#_Toc149114775)

[Лабораторная работа № 06. ЛОКАЛИЗАЦИЯ ИНТЕРНАЦИОНАЛЬНЫХ ДАННЫХ. 15](#_Toc149114776)

[Лабораторная работа № 07. ОБРАБОТКА ТЕКСТА И РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ. 18](#_Toc149114777)

[Лабораторная работа № 08. РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЙ. 24](#_Toc149114778)

[Диаграммы классов 31](#_Toc149114779)

[Классы 33](#_Toc149114780)

[entity 33](#_Toc149114781)

[Ammunition 33](#_Toc149114782)

[Chestplate 34](#_Toc149114783)

[Helmet 34](#_Toc149114784)

[Sword 34](#_Toc149114785)

[Logic 35](#_Toc149114786)

[Manager 35](#_Toc149114787)

[Sorter 35](#_Toc149114788)

[Util 36](#_Toc149114789)

[Creator 36](#_Toc149114790)

[Localization 36](#_Toc149114791)

[Reader 37](#_Toc149114792)

[Writer 37](#_Toc149114793)

[View 38](#_Toc149114794)

[MainWindow 38](#_Toc149114795)

[CreatorWinodw 38](#_Toc149114796)

[TypeSelector 39](#_Toc149114797)

[Код программы 41](#_Toc149114798)

[Ammunition.java 41](#_Toc149114799)

[Chestpalte.java 42](#_Toc149114800)

[Helmet.java 44](#_Toc149114801)

[Sword.java 47](#_Toc149114802)

[Manager.java 48](#_Toc149114803)

[Sorter.java 50](#_Toc149114804)

[Creator.java 51](#_Toc149114805)

[Localization.java 53](#_Toc149114806)

[Logger.java 54](#_Toc149114807)

[Printer.java 54](#_Toc149114808)

[Reader.java 55](#_Toc149114809)

[Writer.java 57](#_Toc149114810)

[Creator.java 59](#_Toc149114811)

[MainWindow.java 62](#_Toc149114812)

[TypeSelector.java 68](#_Toc149114813)

[Ссылка на гитхаб 71](#_Toc149114814)

# Лабораторная работа № 01. МЕТОДОЛОГИЯ РАЗРАБОТКИ TEST DRIVEN DEVELOPMENT

Цель работы: освоить методологию разработки ПО через тестирование (TDD), а также изучить соответствующие тестовые фреймворки для модульного тестирования NUnit и закрепить их практически.

**Задание:**

Необходимо произвести рефакторинг программной системы, созданной в предыдущей лабораторной работе, следующим образом:

* создать отдельный тестовый подпроект в рамках текущего программного решения;
* максимально покрыть основную бизнес-логику модульными тестами;

**Решение:**

Для выполнения лабораторной работы использовалась библиотека Junit.

Открываем структуру проекта Ctrl + Alt + Shift + S -> Libraries -> жмем + (New Project Library) -> выбираем from Maven (рис.1)

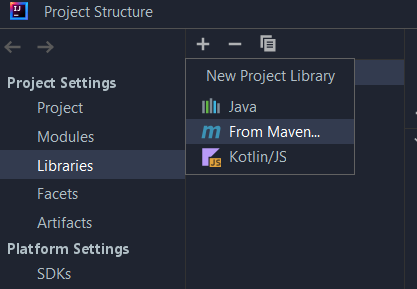


Рис.1.

дальше видим такое окно, в строку поиска вводим “ junit:junit:4.12 ” ждем пока найдет -> OK! -> OK! Должен получиться такой результат (Рис.2)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис.2.

помечен декоратором @Test.

Внутри метода нужно создать ожидаемый результат и получить результат выполнения. Для сравнения используется метод Assert.assertEquals(expected, actual). Если объекты одинаковые, то тест будет обозначен зелёной галочкой.

**Вывод**: Было освоено создание тестов с помощью библиотеки Junit.

# Лабораторная работа № 02. БИБЛИОТЕКИ ПОЛЬЗОВАТЕЛЬСКИЕ

Цель работы: изучить и закрепить на практике создание и использование динамически подключаемых пользовательских библиотек в среде вашей операционной системы.

**Задание:**

Необходимо произвести рефакторинг программной системы следующим образом:

• Вынесите код функций программы в отдельную статическую библиотеку. Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет

результат?

• Вынесите код функций программы в отдельную динамическую библиотеку со статическим вызовом. Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.

• Вынесите код функций программы в отдельную динамическую библиотеку с динамическим вызовом. Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.

• Выполнение второго и третьего задания можно совместить.

• Сравните результаты выполняемых заданий, сделайте вывод.

**Решение:**

Для создания библиотеки нажмём клавиши Ctrl+Alt+Shift+S и попадём в стурктуру проекта. Для создания библиотеки необходимо перейти в Artifcts, нажать + jar->from modules with dependencies(рис.4), откроется окно создания(рис.5).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис.4

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рис.5

В данном окне нужно выбрать пакет, который мы будет добавлять в библиотеку и основной класс, если он существует.

Для сборки библиотеки необходимо нажать Build+Build Artifacts, появится окно со всеми артефактами, где необходимо выбрать нужный нам и нажать build.

После построения в паке проекта появится папка out\artifacts\имя\_библиотеки\_jar. Внутри будет находиться jar файл, который можно запустить либо подключить как библиотеку.

Для подключения библиотеки необходимо перейти в свойства проекта-> Libraries и нажать +->java(рис.8) Появится окно, в котором нужно выбрать путь до jar файла. После подтверждения библиотека будет подтверждена.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 8

Вывод: изучили и закрепили на практике создание и использование динамически подключаемых пользовательских библиотек в среде вашей операционной системы.

# Лабораторная работа № 03. ПОТОКИ ВВОДА-ВЫВОДА

Цель работы: Познакомиться с основами ввода-вывода, структурным шаблоном проектирования «Декоратор», а также практически закрепить данные знания на примере разработки интерактивного приложения.

**Задание:**

В программную систему, разработанную на предшествующем этапе, необходимо внести следующие, расширяющие функционал возможности:

• возможность создавать объекты, которыми манипулирует бизнес логика, различными способами (с использованием генератора случайных чисел и «хардкода») и из различных источников данных (из текстового и бинарного файла) – для этого рекомендуется реализовать иерархию компонентов-создателей и сделать из взаимозаменяемыми в тестовом классе (контроллере);

• возможность выводить результирующие данные не только в консоль, но и сохранять их в файл (текстовый) – для этого рекомендуется также создать иерархию принтеров и сделать их взаимозаменяемыми в классе-контроллере.

По сделанным изменениям должны быть сделаны соответствующие коммиты в Git.

**Решение:**

Для сохранения объектов в текстовый файл была создана своя собственная реализация серриализации, в которой объект записывается в виде [свойство1, свойство2, ...], после чего с помощью библиотеки java.io объект записывается в файл. Для десерриализации объекта из файла считывается весь текст, который разбивается на строки [свойство1, свойство2, ...], после чего эти строки обрабатываются и из них создаются объекты.

Для записи в бинарный файл используется класс ObjectOutputStream из библиотеки java.io. Для того, чтобы объект класса можно было перевести в бинарный код класс должен реализовывать интерфейс Serializable.С помощью метода writeToExcelOnExit он переводится в двоичный код и записывается в файл Excel. Результат показан на рисунке 9.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис.9.

В Excel файл записывается время запуска определенного сохранения, и имя персонажа.

Вывод: Познакомились с основами ввода-вывода, структурным шаблоном проектирования «Декоратор», а также практически закрепить данные знания на примере разработки интерактивного приложения.

# Лабораторная работа № 04. ПОТОКИ ВВОДА-ВЫВОДА

Цель работы: Изучить концепцию и аспекты сериализации и десериализации и закрепить знания и навыки на примере разработки интерактивных приложений.

**Задание:**

В программе, разработанной в предыдущей лабораторной работе, необходимо:

добавить дополнительно два способа инициализации объектов с использованием стандартной и пользовательской («кастомной») сериализации;

попробовать сделать рефакторинг слоя View таким образом, чтобы в контроллере объект-принтер возвращался с помощью объекта-креатора, метод которого брал бы уже готовые объекты из контейнера типа Map. Ключом в данном контейнере может быть тип принтера, а значением – конкретный объект-принтер.

По сделанным изменениям должны быть сделаны соответствующие коммиты в Git.

**Решение:**

Для выполнения задания была добавлена сериализация в json файл. Для этого была подключена библиотека json (инструкция для подключения библиотек лаб. Раб.2)

Для того, чтобы добавить возможность сериализовать объекты в json необходимо подключить библиотеку json.JSONObject.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рис.11

Для записи в файл необходимо создать FileWriter и вызвать метод FileWriter.write(), который преобразует объект в строку, которую затем записываем в файл, как в предыдущих лабораторных работах.

**Вывод:** изучили возможность сериализации в json

# Лабораторная работа № 05. ЖУРНАЛИРОВАНИЕ.

Цель работы: Изучить технологию журналирования в Java.

**Задание:**

Необходимо произвести рефакторинг приложения предыдущей лабораторной работы таким образом, чтобы результирующие данные сохранялись в лог-файл, а также вести журналирование любых исключительных ситуации, которые могут возникнуть в процессе работы программы.

Дополнительно необходимо расширить иерархию принтеров ещё одним компонентом, вывод которого будет базироваться на базе библиотеки log4j.

По сделанным изменениям должны быть сделаны соответствующие коммиты в Git.

**Решение:**

Для журналирования использовалась библиотека log4j. Для создания логера нужно создать файл его конфигурации log4j2.xml(рис.12), в котором будет указана реализация различных видов логов (для информации, ошибок, критических ошибок) и различных их выводов (в консоль, в файл).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис.12.

Для указания пути к файлу конфигурации нужно указать в конфигурации запуска параметр - **Dlog4j.configurationFile=Путь\_к\_файлу.**

Для создания логов нужно получить объект логера:

Logger logger = LogManager.getLogger("имя логера");

Дальше мы можем создавать логи разных уровней:

* info для сообщений
* error для ошибок
* fatal для критических ошибок

# Лабораторная работа № 06. ЛОКАЛИЗАЦИЯ ИНТЕРНАЦИОНАЛЬНЫХ ДАННЫХ.

Цель работы: Изучить технологии локализации и интернационализации данных и применить полученные знания при проектировании и разработки интерактивных приложений.

**Задание:**

Необходимо адаптировать видимый интерфейс программы предыдущей лабораторной работы, а также отдельные её программные элементы (текст, даты, денежные единицы и т.п.) под соответствующую локаль пользователя.

**Решение:**

Для выполнения лабораторной работы использовались библиотеки java.util.Locale для создания перевода; java.util.ResourceBundle для хранения локализаций.

В начале нужно создать несколько файлов ресурсов с расширением .properties (рис. 13)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис.13.

Файлы представляют собой пары ключ-значение. Заполним их одинаковыми ключами, а значения присвоим нужный нам текст(рис. 13-14).

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис.14

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис.15

Далее нужно создать локализацию с помощью *locale*:

*Locale("en", "US")*

Где en – язык, US – страна

Далее нужно создать объект, содержащий локализацию, привязанный к файлу локализации:

*ResourceBundle messages = ResourceBundle.getBundle("messages", currentLocale);*

Далее обращаясь к методу getString этого объекта указывая ключ можно получить значени.

messages.getString(message)

Вывод: Изучили технологии локализации и интернационализации данных и применили полученные знания при проектировании и разработки интерактивных приложений.

# Лабораторная работа № 07. ОБРАБОТКА ТЕКСТА И РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ.

Цель работы: Закрепить навыки работы со строками на примере разработки интерактивных приложений для обработки текстовой информации.

**Задание:**

1. Найти такое слово в первом предложении, которого нет ни в одном из остальных предложений.
2. Из текста удалить все слова заданной длины, начинающиеся на согласную букву.

**Решение:**

класс для описания всего текста

public class TextProcessor {  
  
 public static String processText(String input) {  
 // Заменяем все символы, кроме букв, на звездочки  
 String cleanedText = input.replaceAll("[^a-zA-Z]", "\*");  
  
 // Ваша логика обработки текста  
  
 return cleanedText;  
 }  
  
}

Скриншоты результатов:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис.16

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Рис. 17

Вывод: Закрепили навыки работы со строками на примере разработки интерактивных приложений для обработки текстовой информации.

# Лабораторная работа № 08. РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЙ.

Цель работы: Научиться разворачивать на компьютере конечного пользователя разработанное вами приложение.

**Задание:**

Создать установщик своего приложения.

**Решение:**

Для создания установочного файла необходимо создать exe файл. Для этого используется приложение launch4j(рис. 17)

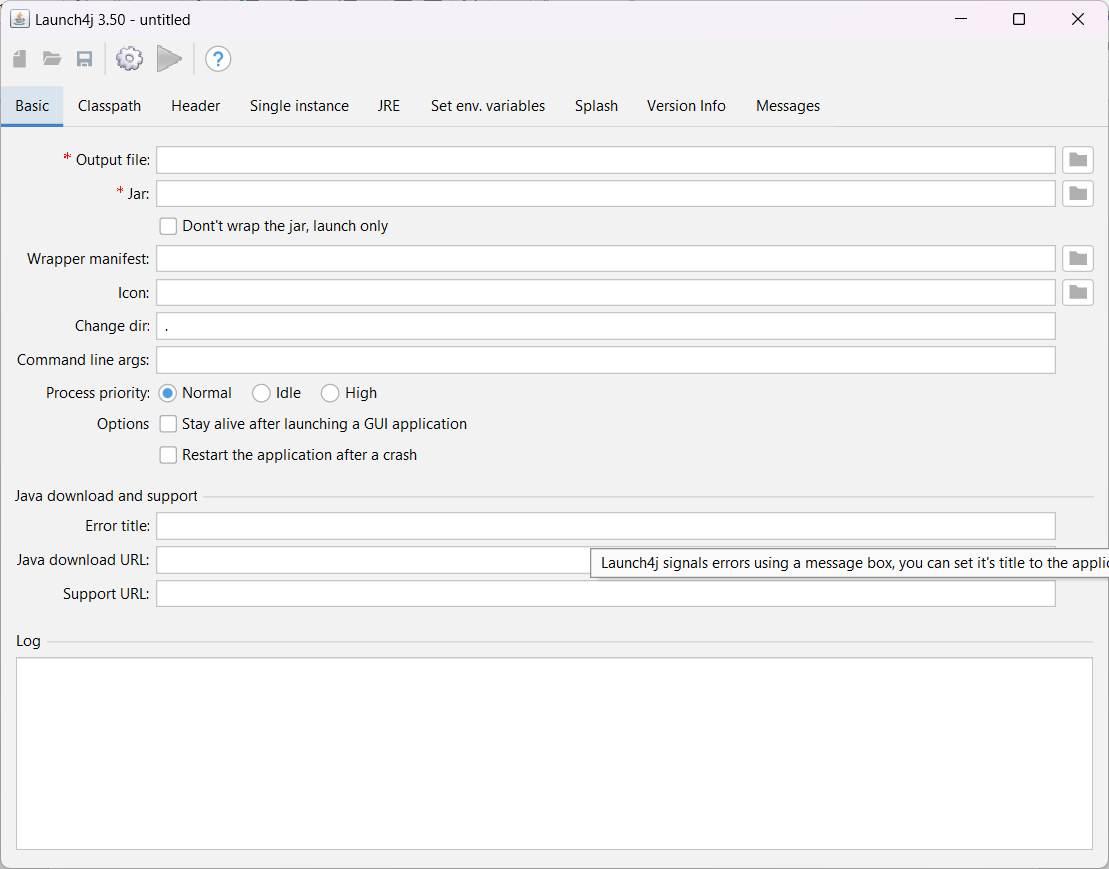


Рис. 17.

Поля:

* Output file - путь к exe файлу
* Jar – путь к jar файлу, который мы будем упаковывать в exe
* Icon – путь к иконе для приложения (**ВАЖНО!** Именно в этом случае нужно чтобы в пути к файлу не было кирилицы, с другими файлами это не требуется)

Далее необходимо нажать на файл шестерёнки и создастся xml файл, который описывает наши параметры. Нажимаем на зелёный треугольник и всё готово.

Для создания установочного файла используется программа Inno Setup Comiler. На стартовом окне приложения нужно создать новый файл (рис.)



Рис.18

В первом окне необходимо указать имя программы, версию, компанию, вебсайт (рис.)

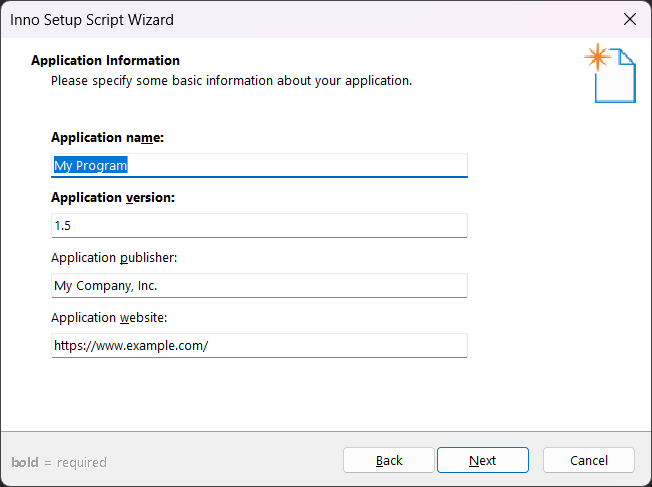
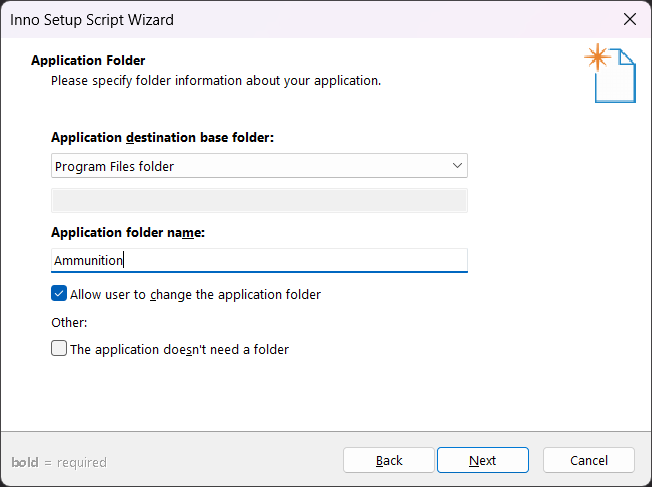


Рис. 19

В следующем окно можно указать начальную папку установки программы(по умолчанию program files), имя папки, разрешить ли менять папку установки, нужна ли в целом папка устновки.

Рис. 20

В следующем окне необходимо указать путь к exe файлу и добавить все необходимые файлы. Имеется два чекбокса: разрешить запустить приложение после установки, у приложения нет исполняемого файла (рис. )

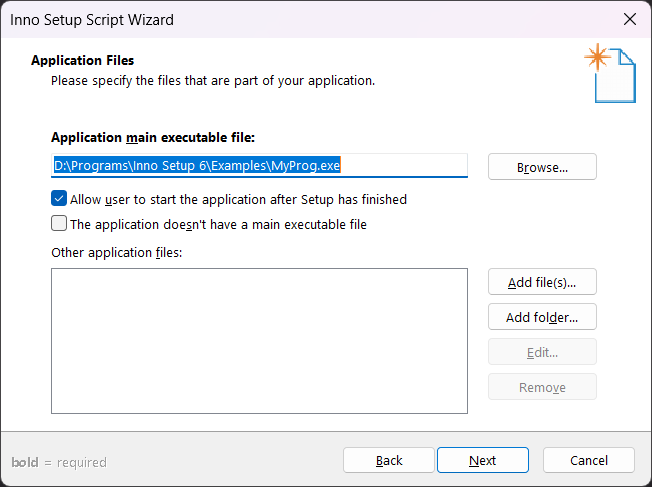


Рис.21

В следующем окне представлен выбор языков программы (рис. ).

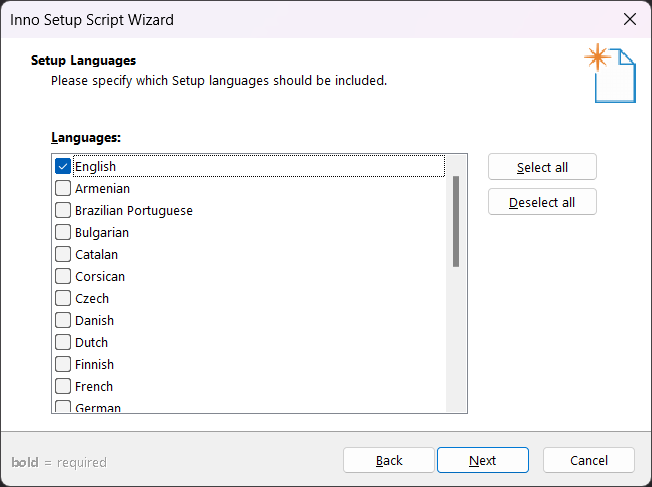


Рис. 22

В следующем окно необходимо указать файл лицензии, файл с информацией перед установкой, файл с информацией после установки.

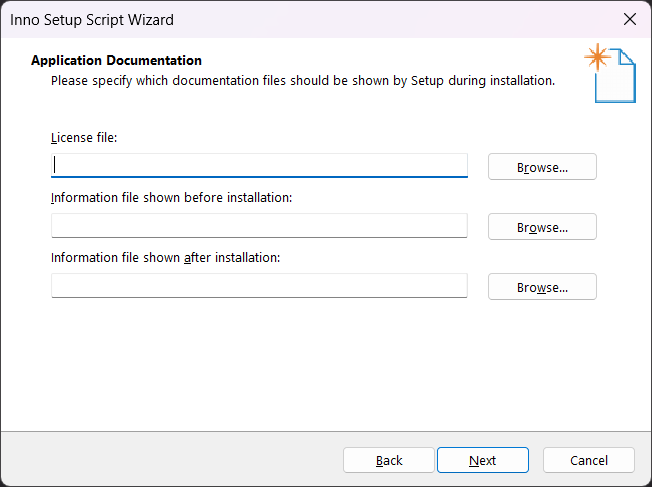
****

Рис. 23

Готово, создан скрипт для создания установочного файла. Далее нужно запустить данный скрипт и будет создан установочный файл.

**Вывод**: Научился разворачивать на компьютере конечного пользователя разработанное приложение.

# Диаграммы классов

Диаграммы классов представлены на рисунках 24-27

Изображение выглядит как снимок экрана, 3D-моделирование, Цифровая сборка, Графическое программное обеспечение

Автоматически созданное описание

Рис. 24

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис.25

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рис. 26

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 27

# Ссылка на гитхаб

https://github.com/cheeselover419/prisontamagotchu