Министерство образования и науки российской федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ПГТУ»)

Кафедра ИиСП

**КУРСОВАЯ РАБОТА**

По дисциплине «Перспективные информационные технологии»

На тему

«Разработка приложения с графическим интерфейсом пользователя»

Выполнил:

Студент ПС-31

Садовин Алексей Альбертович

Проверил:

Доцент кафедры ИиСП

Малов Алексей Николаевич

Дата \_\_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_\_\_\_\_

Йошкар-Ола

2022

Содержаение

[Введение 3](#_Toc94730683)

[Постановка задачи 4](#_Toc94730684)

[Описание и функционал приложения 5](#_Toc94730685)

[Архитектура приложения 6](#_Toc94730686)

[Модель (Model) 6](#_Toc94730687)

[Контроллер (Controller) 7](#_Toc94730688)

[Представление (View) 7](#_Toc94730689)

[Заключение 8](#_Toc94730690)

# Введение

С каждым годом информационные технологии играют все большую роль в жизни людей. Все предприятия, образовательные и государственные учреждения, имеют свою собственную систему, несущую в себе весь объем информации для поддержки прямой деятельности данной организации. В настоящее время существует множество различных информационных систем, каждая из которых имеет свои собственные особенности. Для того чтобы облегчить и ускорить работу людей, данные системы и приложения должны быть построены на одинаковых принципах. При этом временные затраты на адаптацию к работе в данных информационных системах для неподготовленного человека должны быть минимальны.

Актуальность данной темы состоит в том, что пользователь видит и взаимодействует именно с интерфейсом и именно по нему, он оценивает и само приложение, а иногда и весь продукт.

# Постановка задачи

Требуется разработать с использованием паттерна MVC, MVP либо MVVM приложение с графическим интерфейсом пользователя, визуализирующим график суммы гармонических колебаний определенных частот, фаз и амплитуд. Пользователь может добавлять и удалять гармонические функции, а также редактировать значения их частоты, фазы, амплитуды, а также осуществлять выбор гармонической функции (синус либо косинус).



# Описание и функционал приложения

**В приложении есть 3 основные области:**



1. Список всех гармноческий функций, где можно выбрать функцию для изменения или удаления
2. Область для редактирования выбранной гармонической функции
3. Область для визуализации графика суммы гармонических колебаний. Также есть возможность переключения между представлениями значений функции в виде графика и в виде таблицы.

**Также есть 2 кнопки:**

1. Открывает новое диалоговое окно в котором можно добавить новую гармоническую функцию
2. Удаляет выбранную гармоническую функцию

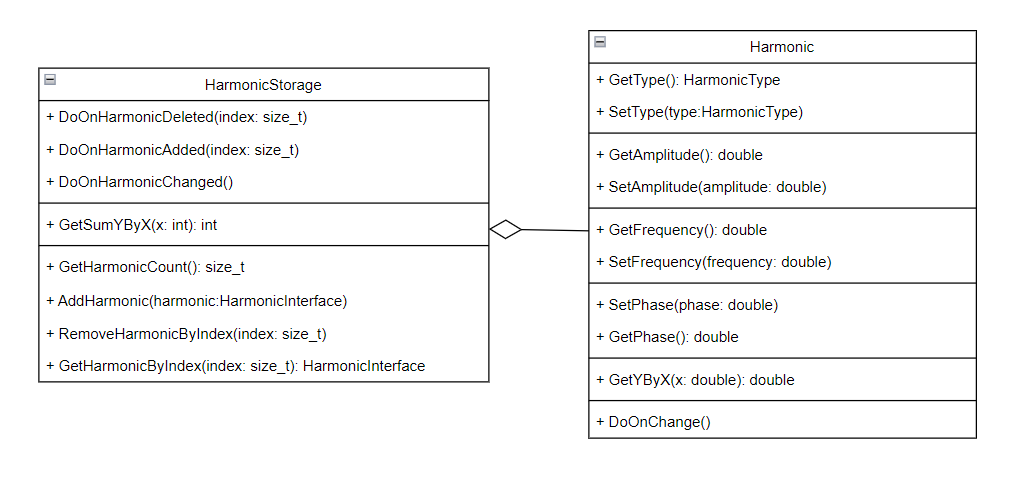
# Архитектура приложения

Архитектура приложения построена на основе паттерне MVC. Есть 3 слоя Model, View, Controller. Модель в приложении активная, т.е она сама уведомляет слушателей о своем изменении.

## Модель (Model)

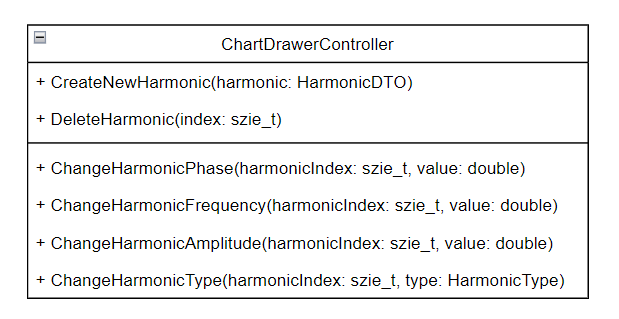
Model состоит из двух сущностей **Harmonic** и **HarmonicStorage**.

* **Harmonic** - сущность модели которая реализует гармонические функции. Он имеет амплитуду, частоту, фазу, а так же тип гармонический функции(sin, cos). Может уведомлять слушателей о своем изменении.
* **HarmonicStorage**  – сущность модели которая хранит гармонические функии. Может уведомлять слушателей о своем изменении.



## Контроллер (Controller)

Контроллер обрабатывает запросы на изменение модели.



## Представление (View)

Представление отвечает за визуалицию модели, получает данные на прямую из модели, подписываясь на ее изменения. Польователь может взаимодействовать с ее элементами, но если какое-то событие должно изменить модель, представление сообщит о действиях пользователя контроллеру, а контроллер в свою очередь изменит модель.

Представление состоит из 5 классов. Главного **ChartDrawerView** и 4 его дополняющих **ChartGraphicView**, **CreareNewHarmonicView**, **HarmonicList**, **HarmonicEditor**.

1. **ChartDrawerView**  - гланый класс представления, который получает основные уведомления от модели.
2. **ChartGraphicView** – визуализирует график суммы и таблицу. Он сам запрашивает данные у HarmonicStorage, чтобы получить нужные значения для отображения графика суммы.
3. **CreareNewHarmonicView** –отвечает за получение данных для создания нового графика.
4. **HarmonicList** – отображает графики хранящиеся в HarmonicStorage
5. **HarmonicEditor –** получает данные для изменения выбранного графика функции

# Заключение

В ходе выполнения курсовой работы было разработано приложение с графическим интерфейсом пользователя, которое позволяет визуализировать график суммы гармончиских колебаний.

Во время работы удалось поработать с таким паттерном, как MVC. Это паттерн, который позволяет отделить логику от визуальной части приложения.