**1. Пояснения к диаграмме классов**

**1.1. Слой сущностей (Entities)**

Классы, отражающие таблицы БД:

* **User**: Пользователь (студент, преподаватель, администратор).
* **Subject**: Учебный предмет.
* **Grade**: Оценка с параметрами (вес, тип, дата).
* **Attendance**: Посещаемость.

**Связи:**

* Один пользователь (User) может иметь множество оценок (Grade) и записей посещаемости (Attendance).
* Один предмет (Subject) содержит множество оценок (Grade).

**1.2. Data Access Layer (DAL)**

* **IGradeRepository**: Интерфейс для работы с оценками.
* **GradeRepository**: Реализация для PostgreSQL (использует Dapper).

**Методы:**

* AddGrade() – добавление оценки в БД.
* GetGrades() – получение оценок студента по предмету.

**1.3. Business Logic Layer (BLL)**

* **GradeService**: Содержит логику расчётов:
  + CalculateWeightedAverage() – средний балл с учётом весов.
  + PredictFinalGrade() – прогноз итоговой оценки.
* **AttendanceService**: Анализ посещаемости.

**1.4. API Layer (Controllers)**

* **GradesController**: Преобразует данные для UI.
  + GetChartData() – возвращает структуру ChartData (даты и значения оценок).

**1.5. Presentation Layer (WinForms)**

* **MainForm**: Главная форма с дашбордом.
  + Использует GradesController для получения данных.
* **ChartData**: DTO (Data Transfer Object) для передачи данных в графики.

**2. Ключевые связи и зависимости**

1. **Агрегация (o-->)**:
   * GradeService зависит от IGradeRepository.
   * GradesController зависит от GradeService.
2. **Реализация интерфейсов (..|>)**:
   * GradeRepository реализует IGradeRepository.
3. **Композиция (\*--)**:
   * User содержит список Grade и Attendance.

## ****3. Дополнения****

### ****3.1. Паттерны в архитектуре****

* **Repository** – для абстракции работы с БД.
* **Dependency Injection (DI)** – внедрение IGradeRepository в GradeService.
* **DTO** (ChartData) – для передачи данных между слоями.

### ****3.2. Расширение диаграммы****

Если нужно добавить:

* **Отчёты**: Класс ReportService с методом GeneratePdf().
* **Кеширование**: Класс CacheManager (например, с Redis).
* **Логирование**: Интерфейс ILogger и его реализация.