РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ «ТРЕКЕР ПРИВЫЧЕК»

Габдуллин Д.Р., студент
Гилёв А.Ю., старший преподаватель
Бирский Филиал УУНиТ, г. Бирск, Россия

Аннотация. В статье рассматривается проектирование структуры учебного приложения в соответствии с современными практиками. Основная задача - разумное выделение классов и разделение обязанностей между ними. Приведены основные классы приложения и диаграмма классов.

Ключевые слова: Проектирование программного обеспечения, Принципы SOLID, шаблоны проектирования.

Введение

В рамках изучения дисциплины «Проектирование информационных систем», мы поставили перед собой задачу разработки архитектуры небольшого учебного приложения с применением лучших практик разработки больших и сложных информационных систем.

Приложение «Трекер привычек» предназначено для напоминания пользователю о необходимости периодически выполнять некоторые действия, которые помогут сформировать привычки. Надеемся, что только полезные.

Далее под «привычкой» будет пониматься задача для приложения, которую оно должно периодически предоставлять пользователю.

Архитектура «приложения «Трекер привычек»

Архитектура приложения «Трекер привычек» включает несколько ключевых компонентов:

- Класс Habit – основная сущность, представляющая привычку пользователя.
- Классы Progress, Reminder, Achievements и другие вспомогательные классы, которые помогают управлять состоянием привычки, напоминанием и достижениями соответственно.
- Контроллер HabitsController, который управляет списком привычек и их жизненным циклом (добавление, редактирование, удаление).
- Репозиторий Repository, отвечающий за хранение данных о привычках.

Описание основных классов

Класс Habit

Этот класс представляет собой основную единицу отслеживания привычек. Он содержит следующие свойства и методы:

- Свойства:
  - Name - название привычки,
  - Description - описание привычки,
  - Remind - объект класса Reminder,
  - Progress - объект класса Progress.
- Конструктор - принимает параметры для инициализации объекта Habit.
- Метод UpdateReminder - обновляет настройки напоминания для данной привычки.

Класс Progress

Этот класс хранит информацию о прогрессе выполнения привычки. Основные элементы включают:

- Свойство State - текущее состояние выполнения привычки (true/false).
- Конструктор, принимающий начальную дату, интервал повторения, состояние и заметки.
- Методы:

- Свойство EndDate, которое вычисляется автоматически по начальной дате и интервалу.
- Метод ClearProgress, предназначенный для сброса прогресса.

Класс Reminder

Этот класс отвечает за управление напоминаниями о выполнении привычки. В нем есть такие компоненты:

- Свойства:
  - FrequencyInDay: частота напоминаний в течение дня.
  - WhichDayRemind: день недели, когда должно быть отправлено напоминание.
- Методы:
  - Конструктор: создает объект Reminder с заданной частотой и днем напоминания.
  - Событие RemindAlarm: генерируется при срабатывании напоминания.
  - Установка нового напоминания (SetReminder),
  - Удаление существующего напоминания (DeleteReminder)
  - Метод уведомления пользователя (Notify).

Класс HabitsController

Это центральный компонент управления списками привычек. Его задачи включают добавление, изменение и удаление привычек, а также расчет статистик:

- Свойство:
  - habitList – список всех привычек.
- Методы добавления, изменения и удаления привычек:
  - Добавление новой привычки (AddHabit);
  - Редактирование существующей привычки (EditHabit);
  - Удаление привычки (DeleteHabit).

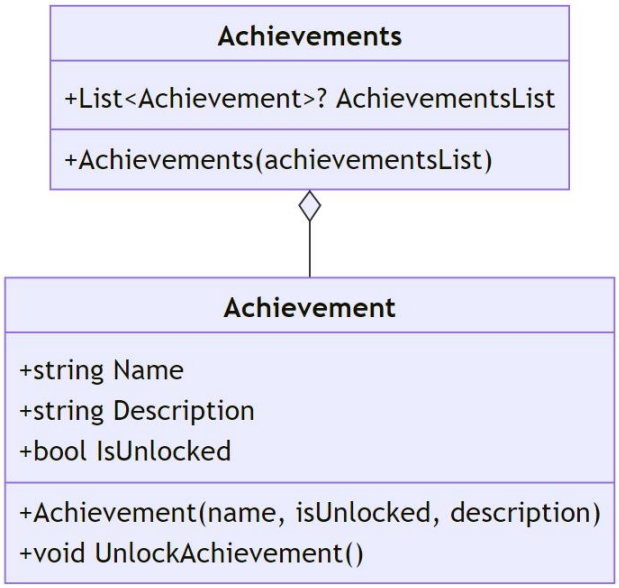
- Методы расчета статистики:
  - Расчет успешности выполнения привычек (CalculateSuccessRate);
  - Подсчёт непрерывных серий выполнения привычек (CalculateStreaks);
  - Определение количества пропущенных дней (CalculateMissedDays).
- Дополнительные функции:
  - Возможность поделиться прогрессом (ShareProgress);
  - Соединение с сообществом пользователей (ConnectToCommunity);

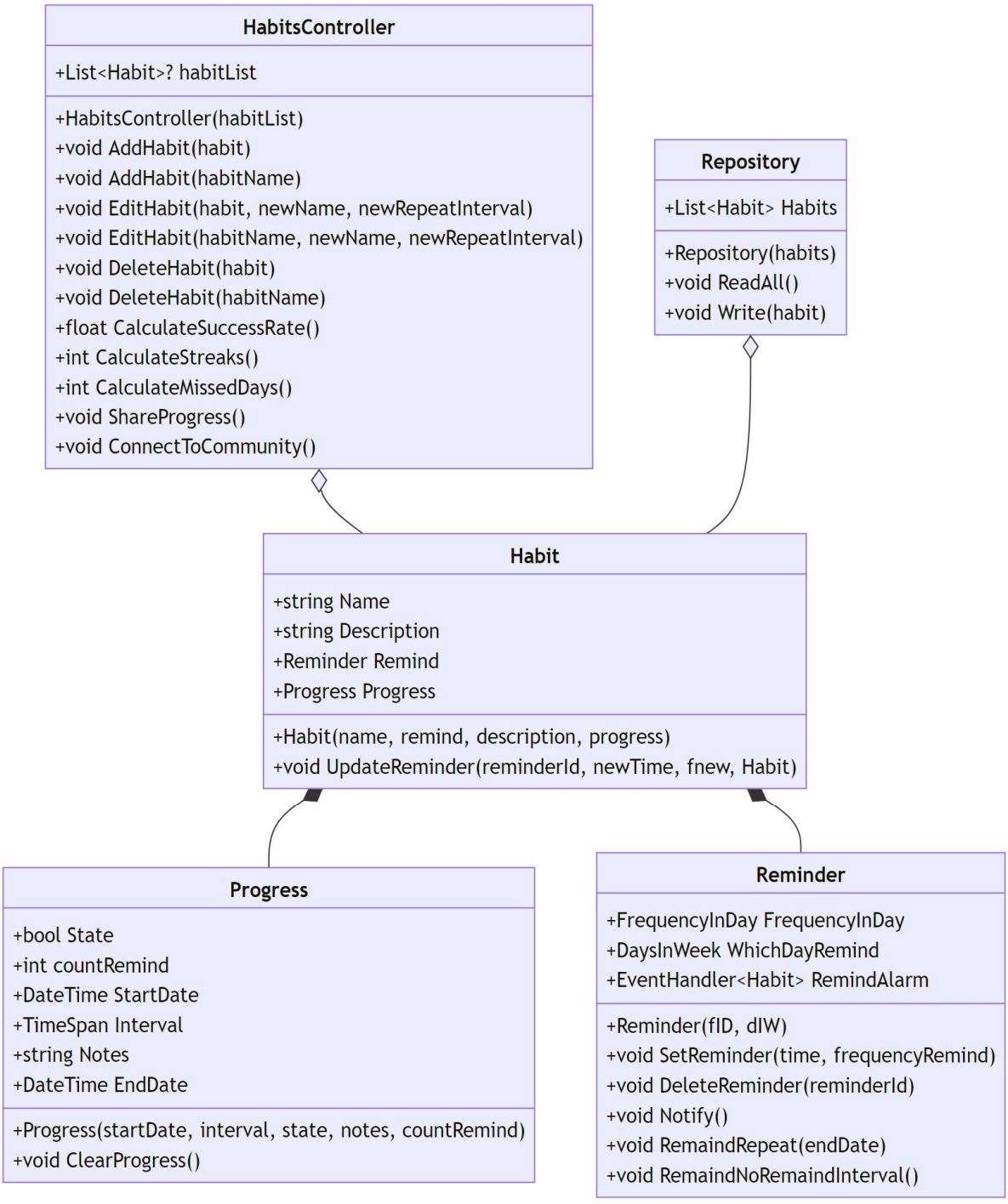
Репозиторий Repository

Этот компонент предназначен для хранения и загрузки данных о привычках:

- Поле Habits – коллекция объектов Habit, представляющих все привычки.
- Метод ReadAll – загружает данные о привычках.
- Метод Write – сохраняет текущее состояние привычки.

Диаграмма основных классов приложения





На этой диаграмме видно, что:

1. Habit имеет ассоциации с объектами Progress и Reminder.
2. HabitsController управляет коллекцией объектов Habit.
3. Repository хранит коллекцию объектов Habit.

Пример использования описанных классов

Разработанные классы далее применяются в приложении с библиотекой WindowsForms

```
public MainForm()
{

    InitializeComponent();

    // Инициализация потока для обновления состояния
    Thread updaterthrd = new Thread(Updater);
    updaterthrd.Start();

    // Начальный список привычек (пустой) и контроллер привычек
    List<Habit> habitList = new List<Habit> ();
    HabitsController hController = new HabitsController(habitList);

    // создание тестовой привычки
    hController.AddHabit(
        new Habit("тестовая привычка",
            new Reminder(FrequencyInDay._5m, DaysInWeek.Monday),
            "описание",
            new Progress(new DateTime(2015, 7, 20),
            new TimeSpan(2),
            false,
            "заметка",
            0)));

    //считать данные с файла
    List<Habit> forRepos = new List<Habit>();
    Repository repos = new Repository(forRepos);
    repos.ReadAll();
}
```

В этом примере кода демонстрируется процесс создания и запуска формы Windows Forms, а также инициализация некоторых компонентов приложения «Трекер привычек»:

- *Создание списка привычек и контроллера*. Здесь создается пустой список habitList, который затем передается в конструктор HabitsController. Контроллер hController будет управлять этим списком привычек.
- *Добавление тестовой привычки*. В контроллер добавляется новая привычка с именем «тестовая привычка». При этом создаются объекты Reminder и Progress с соответствующими параметрами. Напоминание настроено на частоту 5 минут в понедельник, а прогресс начинается с 20 июля 2015 года с интервалом в 2 часа.

- *Чтение и запись данных из хранилища*. Сначала создается еще один пустой список привычек forRepos, который используется для создания экземпляра репозитория repos. Затем вызывается метод ReadAll(), чтобы считать данные о привычках из хранилища.

Выводы

Архитектура приложения «Трекер привычек» четко разделяет обязанности между компонентами: класс Habit описывает привычки, HabitsController управляет ими, Repository хранит данные, а Progress и Reminder отслеживают выполнение и напоминают о привычках. Подобный подход при проектировании сложных программ обеспечивает модульность, расширяемость и относительную простоту тестирования приложения.

Литература

1. Арораа Г., Чилберто Д. Паттерны проектирования для C# и платформы .NET Core. — СПб.: Питер, 2021. — 352 с.

2. Вайсфельд М. Объектно-ориентированное мышление. — СПб.: Питер, 2014. — 304 с.

3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. паттерны проектирования. — СПб.: Питер, 2018. — 368 с.

4. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. — СПб.: Питер, 2018. — 352 с.

5. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#. – СПб.: Символ-Плюс, 2011. – 768 с.

6. Тепляков С. Паттерны проектирования на платформе .NET. — СПб.: Питер, 2015. — 320 с.