Дисциплина «Программирование»

Практическое задание №7

9 февраля – 22 февраля

«Управление задачами»

Процесс выполнения этого задания состоит из трёх частей:
1) реализация программы, согласно описанным в условии требованиям;
2) оценивание работ других студентов;
3) период «споров».

Период реализации программы:

После выдачи задания Вам необходимо выполнить его и загрузить архив (*.zip) с решением задачи (полностью заархивировать решение, созданное средой разработки) до крайнего срока. Работа должна выполняться студентом самостоятельно. В работе строго запрещается указывать ФИО, а также любую другую информацию, которая может выдать авторство работы. В случае выявления факта деанонимизации работы, работа может быть аннулирована.

Период взаимного оценивания:

После окончания срока, отведенного на реализацию программы, начинается период взаимного оценивания. Вам будет необходимо проверить пять работ других студентов, также выполнявших данное задание, согласно критериям оценивания. Проверка осуществляется анонимно: Вы не знаете, чью работу Вы проверяете, также, как и человек, кому принадлежит решение, не знает, кем была проверена его работа. Помимо оценки Вам необходимо указать комментарий к каждому из критериев. В случае, если Вы снижаете балл, необходимо подробно описать, за что именно была снижена оценка. Также рекомендовано писать субъективные комментарии, связанные с тонкостями программной реализации, предлагать автору работу более оптимальные на ваш взгляд решения. Снимать баллы за субъективные особенности реализации запрещено.

Период споров:

По окончании периода взаимного оценивания, в случае если Вы не согласны с оценкой, выставленной одним из проверяющих, Вы можете вступить с этим студентом в анонимный диалог с целью уточнения причин выставления оценки по тому или иному критерию. В случае, если проверяющий не ответил Вам, или вы не пришли к обоюдному решению об изменении оценки, Вы можете поставить флаг, и работа будет рассмотрена одним из преподавателей.

Флаги, поставленные без предварительного обсуждения с проверяющим или после окончания периода выставления флагов, будут отклонены.

Также допустима перепроверка Вашей работы преподавателем или ассистентом. В таком случае оценки других проверявших работу не учитываются.

Дедлайн загрузки работы: 22 февраля 23:59
Дедлайн проверки: 25 февраля 23:59
Дедлайн обсуждения оценок: 27 февраля 23:59
Дедлайн выставления флагов: 28 февраля 23:59

Возможность поздней сдачи работы в этом задании предоставляться не будет.

Оценивание:
$О_{итог} = 0,8 * О_{задание} + 0,2 * О_{проверки}$,
где $О_{задание}$ – неокруглённая десятибалльная оценка за решение задания выставленная проверяющими с учётом возможной перепроверки преподавателем, а $О_{проверки}$ неокруглённая десятибалльная оценка, выставленная студентами, чьи работы вы проверяли. Также в случае, если преподаватель обнаружит, что Вы проверили работы некачественно к Вам могут быть применены санкции в виде штрафа до 3 десятичных баллов от $О_{итог}$ (в таком случае оценка вычисляется по формуле $О_{итог} = О_{задание} – Штраф$).

Необходимо разработать библиотеку классов – «Управление задачами» + консольный интерфейс, использующий разработанную библиотеку.

В библиотеке классов должны присутствовать следующие сущности:

Проект - служит для группировки задач. У каждого проекта должны присутствовать следующие поля (свойства): название проекта и список задач проекта. Список задач может меняться - можно создавать новые задачи и удалять ненужные. Максимальное количество задач в проекте ограничено определенным целочисленным значением.

Задачи - могут быть разного типа и иметь подзадачи. У каждой задачи должны присутствовать следующие поля (свойства): название задачи и дата ее создания.
Также должен присутствовать статус, который позволяет определить, что происходит с задачей в данный момент.

Виды статусов:

1. Открытая задача - статус по умолчанию.
2. Задача в работе.
3. Завершенная задача.

Типы задач:

1. Тема, или Epic - большая задача, для реализации которой нужно много времени, включает в себя несколько *меньших* подзадач.
2. История, или Story - задача, по объему работ меньшая, чем Epic. Может быть подзадачей Epic.
3. Задача, или Task - задача, по объему работ меньшая, чем Story. Может быть подзадачей Epic.
4. Ошибка, или Bug - задача, описывающая проблему в проекте. Никак не связана с Epic.

Поля и функциональные члены класса, общие для всех типов задач, должны быть вынесены в базовый (родительский) класс. Допустимо добавлять другие члены классов.

Задачам типа Task и Bug может быть присвоен один исполнитель, а задачам типа Story - несколько. Одним из возможных способов реализации этого функционала является то, что классы, описывающие перечисленные выше типы, должны реализовывать интерфейс **IAssignable**, определяющий свойство - список пользователей-исполнителей задачи и методы, позволяющие работать с этим списком. Данный функционал необходимо реализовать по вашему усмотрению (но помните, что реализация через интерфейс **IAssignable**, как наиболее правильная с точки зрения ООП, относится к дополнительному функционалу).

Пользователь - человек, который может быть назначен исполнителем задачи. У каждого пользователя должно присутствовать поле (свойство) - его имя.

Задание:

Используя описанную выше библиотеку классов разработать консольное приложение (.Net Framework 4.7.2 (или 4.8), .NET Core 3.1 или .NET 5) со следующей функциональностью:

1. **Работа с пользователями.** Создание пользователя. Просмотр списка пользователей. Удаление пользователя.
2. **Работа с проектами.** Создание проекта. Просмотр списка проектов (помимо названия для каждого проекта должно отображаться количество связанных с ним задач). Изменение названия проекта. Удаление проекта.
3. **Работа с задачами в проекте.** Добавление новой задачи в проект. Назначение исполнителей из списка пользователей для выполнения задачи. Изменение исполнителей задачи. Удаление исполнителей из

задачи. Изменение статуса задачи. Просмотр списка задач (помимо названия для каждой задачи должна отображаться дата ее создания, имена исполнителей и статус). Группировка задач по статусу. Удаление задач из проекта.

4. **Для задач типа Epic**: назначение подзадач, удаление подзадач.
5. **Сохранение состояния приложения.** Автоматическое сохранение всех данных (проекты, задачи, пользователи) в папку (или файл) при закрытии приложения. Таким образом, при закрытии приложения данные не должны быть потеряны. Загрузка сохраненных данных при повторном запуске программы. Формат данных определяется разработчиком.
6. **[Дополнительный функционал]** Реализовать ограничение количества исполнителей у задач типа Task, Bug или Story с использованием интерфейса **IAssignable**.
7. **[Дополнительный функционал]** Аналогично разработать оконное приложение Windows Forms App (.Net Framework 4.7.2 (или 4.8), .NET Core 3.1 или .NET 5).

Дополнительные требования (критерии оценивания):

1. Для проверки в систему PeerGrade должен быть загружен архив с решением. Ожидается, что проверка работы будет проводиться, в среде разработки Visual Studio 2019, поэтому в случае выполнения задания с использованием другой среды разработки настоятельно рекомендуется проверить возможность открытия и запуска проекта в этой среде разработки.
2. Текст программы должен быть отформатирован согласно кодстайлу языка C#[1]. Для автоматического форматирования в среде Visual Studio достаточно нажать Ctrl+K, D. Идентификаторы должны соответствовать соглашению о именовании C#[2]. Основным требованием, которое требуется соблюдать в данном задании – это написание имён локальный переменных с использованием camelCasing, а при именовании методов и типов PascalCasing.
3. Программа не должна завершаться аварийно (или уходить в бесконечный цикл) при любых входных данных. При некорректных входных данных программа должна выводить сообщение об ошибке и запрашивать ввод заново.
4. Программа должна быть декомпозирована. Каждый из логических блоков должен быть выделен в отдельный метод. Не строго, но желательно, чтобы каждый метод по длине не превышал 40 строк.

---

[1] https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

[2] https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/naming-guidelines

5. Интерфейс программы должен быть понятен. Пользователю должны выводиться подсказки о возможных дальнейших действиях и иные необходимые сообщения. Предполагается, что для успешного использования программы не требуется обращения к исходному коду программы.
6. Текст программы должен быть документирован. Необходимо писать, как комментарии перед методами, так и комментарии, поясняющие написанный внутри метода код. Названия переменных и методов должны быть на английском языке и отражать суть хранимых значений / выполняемых действий.
7. Работа должна быть выполнена в соответствии с заданием, критерии оценивания (“Rubric”) доступны для предпросмотра в системе “PeerGrade”.
8. Для получения отличной оценки более 8 студенту необходимо реализовать дополнительный функционал, описанный в п.6 – п.7 задания.
9. Также оценивается общее впечатление, которое производит работа (как с точки зрения пользовательского интерфейса, так и с точки зрения написания кода программы). Эта часть оценки остаётся на усмотрение проверяющего.