Контрольное домашнее задание

Результатом выполнения лабораторной работы является архив, содержащий исходный код программы на языке C#, решающий задачи, поставленные в рамках задания

Дата сдачи работы:

До 2020-03-11 11:00

Даты отправки рецензии на работы других студентов:

До 2020-03-15 11:00

Порядок сдачи работы

Архив с выполненным заданием быть отправлен в виде вложения в электронном письме на ящик peerrobot@ithse.ru с почтового ящика студента в домене **edu.hse.ru**.

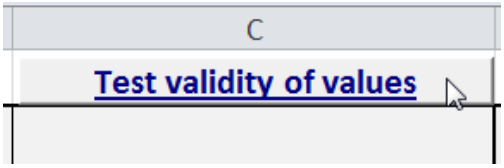
Требования к письму с выполненным заданием:

1. Zip-архив выполненного задания с названием KDZ.zip высылается вами **ответным** письмом на письмо с заданием (на тот же адрес, с которого пришло задание с той же темой, без ручных изменений).
2. Архив выполненного задания должен быть анонимизирован, то есть в названиях программы, коде и тексте программы не должно содержаться информации об авторе.
3. **Во всех проектах** вашего решения необходимо удалить все файлы из папок **bin** и **obj**.
4. Проверьте, что в конце темы вашего ответного письма сохранился уникальный идентификатор {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}

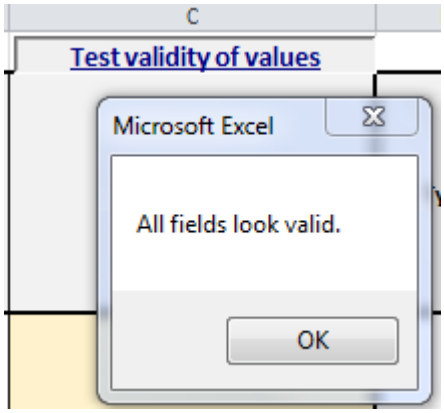
Порядок проверки работы

1. Работы для проверки в анонимизированном виде рассылаются на почтовые адреса студентов в домене **edu.hse.ru**. Каждое письмо содержит 2 файла:
    a. Архив с исходным кодом.
    b. Файл с пустой проверочной формой (xlsx-файл).
2. Каждый студент оценивает пять работы однокурсников по приложенной проверочной форме (то есть получает пять писем).
    a. Если студент не проверит хотя бы одну работу, из итоговой оценки за выполнение задание вычитается 2 балла.
3. Результат проверки каждого из решений (в виде заполненной проверочной формы) должен быть отправлен в виде вложения в ответном письме.
    a. Одна проверка – одно письмо. Убедитесь, что идентификаторы в заголовке письма, на которое вы отвечаете, и файла-проверки совпадают.
    b. Письма должны быть присланы в установленный период проверки работ! Письма, присланные вне периода проверки, не рассматриваются!

Открываете оценочный файл из письма и работу из письма (важно, чтобы номера соответствовали). Проверяете работу по критериям из оценочного листа и выставляете оценку в столбец, выделенный желтым цветом. Проверяете правильно ли заполнена форма, нажав на кнопку сверху:



Если все хорошо, то выведется сообщение:



Если все верно, то сохраняете файл, обязательно оставляя исходное название (например, **`Review 8896706.xlsm`**) и отправляете в ответ на письмо, которое получили.

Требования к письму с результатом проверки задания однокурсника:

1. Файл с проверочной формой НЕ архивируется и должен быть единственным вложением письма.

Имя файла с проверочной формой должно остаться неизменным

Задание

Вариант 4

Реализуйте программу с интерфейсом на WinForms/WPF, которая позволяет:

1. Считать данные об игровых юнитах из файла Dota2.csv

   Каждая строка представляет информацию об одном юните. В таблице с данными хранятся следующие параметры юнитов:

   1. name
   2. type
   3. baseStr
   4. baseAgi
   5. baseInt
   6. moveSpeed
   7. baseArmor
   8. minDmg
   9. Regeneration

   Также необходимо добавить расчет полей Health и MaxHealth, равные *baseStr * 29* для данного юнита.

   NB: для работы с файлом Dota2.csv необходимо реализовать собственный парсер. Встроенную реализацию CsvReader и пр. использовать нельзя.

2. Вывести информацию из файла на экран приложения с возможностью ее редактирования. Рекомендуется использовать DataGridView.
3. Отфильтровать информацию на экране приложения по следующим параметрам юнитов: type, moveSpeed, regeneration
4. Запустить процесс сражения между двумя игроками, где Игрок_2 - компьютер:
    1. Игрок_1 должен иметь возможность по выведенной информации из файла выбрать героя;
    2. Игрок_2 должен иметь возможность по выведенной информации из файла выбрать героя, отличного от выбранного Игроком_1;
    3. Процесс сражения делится на раунды. За один раунд оба игрока выполняют следующие действия:
        i. Игроки производят процесс нападения:
            1. Игрок_1 выбирает:
                a. Убегать
                b. Защищаться
                c. Атаковать
            2. Игрок_2 выбирает действие из тех же вариантов;
            3. Если оба игрока убегают, то с вероятностью 0.2 игроки (независимые вероятности) юнит восстанавливает здоровье в размере *5 * Regeneration.* Необходимо учесть, что здоровье юнита после восстановления не должно превысить максимальное допустимое здоровье;
            4. Если убегает только один игрок, то убегающий юнит теряет 1% максимального здоровья. Необходимо учесть, что уровень здоровья должен остаться не меньше 2 ед.;
            5. Когда оба игрока защищаются, то ничего не происходит;
            6. Если хотя бы один из игроков атакует, то:
                a. Для каждого юнита считается показатель по формуле: minDmg * baseStr / 10 + baseArmor * baseAgi / 10;
                b. Игрок, чей юнит набрал больше очков, наносит урон (если он выбрал атаковать) в размере minDmg * baseStr / 20;
            7. Раунд заканчивается. Следующий раунд начинается с выбора действия Игроком_1.

ii. Проигрывает тот игрок, чей юнит лишается очков здоровья раньше противника.

5. Реализовать возможность автоматического сохранения текущего сражения после каждого раунда в XML файл, разработанного Вами формата. При запуске программы предлагать пользователю продолжить последнее сохраненное сражение (если оно есть) или начать новое сражение.

NB: необходимо написать собственный код, использовать механизмы сериализации нельзя. Рекомендуется использовать класс XmlDocument.

Замечания:

1. Необходимо соблюдать все изученные принципы ООП
2. Необходимо соблюдать декомпозицию (включая разделение классов по файлам и создания библиотек классов)
3. Явной архитектуры и спецификации нет, необходимо продумать самому то, как будет устроена ваша программа. Рекомендуется сперва попробовать изобразить работу программы на рисунке.

Для особо желающих:

1. Изучив LINQ, часть задач из КДЗ могут решиться проще.

Замечания:

4. Необходимо соблюдать инкапсуляцию
5. В классах можно добавлять свои свойства для доступа к полям, методы и поля
6. Можно добавлять свои классы/члены классов
7. Спецификацию полей и методов типов из условия менять нельзя

Ограничения и требования:

1. Предусмотреть цикл повторения решения
2. Использовать конструкцию **try catch** в местах, где могут возникнуть исключения. Максимально конкретизировать реакцию программы по типам исключений.

Существенные требования:

1. Программа должна компилироваться.
2. Входные данные должны обрабатываться и не порождать исключительные ситуации.
3. Не изменять спецификацию указанных в задании нестатических методов.

Требования к интерфейсу:

Интерфейс должен быть реализован на WindowsForms или WPF (только .Net Framework).