Дисциплина «Программирование»

Практическое задание №1

11 января – 25 января

«Транспорт»

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

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. 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 четыре работы других студентов, также выполнявших данное задание, согласно критериям оценивания. 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.

Дедлайн загрузки работы: 18 января 23:59
Дедлайн проверки: 24 января 23:59
Дедлайн обсуждения оценок: 25 января 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 и консольное приложение согласно описанному заданию.

Описание задания:

В библиотеке классов "**EKRLib**" описать классы **Transport**, **Car**, **MotorBoat** и **TransportException**. Абстрактный класс **Transport** описывает абстрактное транспортное средство и содержит (можно создавать дополнительные члены для всех классов, но *нельзя определять конструктор без параметров*):
1) Строковое свойство **Model** (модель). Модель должна состоять только из заглавных латинских символов и цифр и длину ровно 5 символов. При несоответствии выбрасывать исключение типа **TransportException** с сообщением "Недопустимая модель <**Model**>".
2) Целочисленное свойство **Power** (мощность в лошадиных силах, тип uint), не может быть меньше 20 (для некорректных данных выбрасывается исключение типа **TransportException** с сообщением “мощность не может быть меньше 20 л.с.”).
3) Переопределенный метод **ToString()** возвращает строку формата "Model: <Model>, Power: <Power>".
4) Абстрактный метод **string StartEngine()** переопределяется в производных классах для получения звука (в виде строки), издаваемого транспортным средством.
5) Конструктор с параметрами для свойств **Model** и **Power**.
Класс **Car** с параметрическим конструктором, описывающий автомобиль (наследник класса **Transport**). Требования:
1) Переопределенный метод **ToString()**, приписывает слева строку "Car. " к строке, возвращаемой из одноименного метода базового класса.
2) Переопределенный метод **StartEngine**(), возвращающий строку "<Model>: Vroom", где <Model> - модель.
Класс **MotorBoat** с параметрическим конструктором, описывающий моторную лодку (наследник класса **Transport**). Требования:
1) Переопределенный метод **ToString()**, приписывает слева строку "MotorBoat. " к строке, возвращаемой из одноименного метода базового класса.
2) Переопределенный метод **StartEngine**(), возвращающий строку "<Model>: Brrrbrr", где <Model> - модель.
<u>В консольной программе:</u>
1) Создать список (не массив!) с элементами типа **Transport**, состоящий из случайного (в диапазоне [6; 10)) количества элементов и заполнить его ссылками на случайным образом сгенерированные объекты **Car** и **MotorBoat** (выбор типа объекта равновероятен): **Model** и **Power** генерировать случайным образом (**Power**

- в диапазоне [10; 100), **Model** - согласно спецификации). При возникновении исключения – повторить попытку создания объекта (текст сообщения из исключения выводить на экран). После создания объекта (используя **WriteLine**) выводить на экран результат вызова **StartEngine**().
2) Информацию о транспортных средствах (возвращаемую методом ToString()) из сформированного списка записать в текстовые файлы "**Cars.txt**" и "**MotorBoats.txt**" (кодировка utf-16, расположение: папка с файлом решения), содержащие только сведения о машинах (**Cars.txt**) и моторных лодках (**MotorBoats.txt**) соответственно (информация о каждом транспортном средстве – с новой строки).

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 не ниже 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 нашего курса. Для автоматического форматирования в среде Visual Studio достаточно нажать Ctrl+K, D.
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.
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 и критериями оценки.
8. 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.