**Programming BI.**

**Проверяемые навыки**

1. Basic OOP and class composition/aggregation
2. Collections
3. File input-output

**Introduction**

В первом домашнем задании курса по программированию на C # вы внедрите back-end интеллектуальной системы парковки для большого торгового центра.

(1 Термин «back-end» относится к функциональности приложения, которую конечный пользователь не видит, однако он необходим для правильного функционирования всей системы.)

Рассматриваемая система может служить центральным интерфейсом между системами управления воротами с устройствами распознавания автомобильных номеров, киосками оплаты парковки, мобильными приложениями для пользователей и другими компонентами системы.

Давайте сначала опишем типичную парковочную систему с точки зрения пользователя:

1. Посетитель прибывает к одному из въездных ворот, выдается распечатанный билет и разрешается въезд на автомобиле при условии наличия свободных парковочных мест. В нашей интеллектуальной системе парковки есть система распознавания видео, которая считывает номер автомобиля при въезде и выезде с парковки.

2. Посетитель паркует машину в свободном месте.

3. Покинув торговый центр, клиент вставляет парковочный билет в платежный киоск и оплачивает парковку в соответствии с тарифной таблицей.

Предполагается, что таблица тарифов имеет следующий формат (фактические цифры могут отличаться):

(Смотри оригинал в английском файле)

Minutes Rate (RUB) 15 0

60 50 120 100

180 140 240 180 ...

600+ 350

Тарифную политику в приведенной выше таблице следует понимать следующим образом: сеансы парковки продолжительностью до 15 минут являются бесплатными, сеансы продолжительностью до 60 минут стоят 50 рублей и т. Д. Сеансы продолжительностью более 600 минут тарифицируются по максимальной ставке. (последний ряд таблицы). В задании мы не будем рассматривать сложные случаи, когда посетители оставляли свои машины более суток. В любом случае просто возьмите разницу между текущим временем и временем входа (в минутах), даже если между этими двумя годами пройдет один год, вы все равно будете взимать 350 рублей.

4. После совершения платежа посетителю предоставляется дополнительное время, чтобы покинуть парковку, мы будем называть это «бесплатным периодом выхода». Чтобы логически согласовать это с таблицей тарифов, в качестве значения из первой строки можно взять период свободного выхода (15 минут в таблице выше), этот период отсчитывается от времени оплаты, а не от времени входа!

5. Выйдя из парковки, посетитель вставляет билет на выездных воротах в специальный автомат; если автомобиль выезжает в течение разрешенного свободного времени после оплаты, ворота открываются. В противном случае в выходе отказано, и клиент должен заплатить дополнительную плату согласно тарифной таблице.

Ниже приведены некоторые примеры (предполагается, что все времена находятся в пределах одного дня, тарифная таблица сверху взята для расчета):

Сценарий 1:

1. Автомобиль въезжает на парковку в 10:05:10

2. Оплата производится в 11:05:11, заряд = 100 руб. 3. Автомобиль выезжает с парковки в 11:10:00, выезд предоставляется

Сценарий 2:

1. Автомобиль въезжает на парковку в 12:10:00

2. Оплата производится в 12:16:45, заряд = 0 руб. 3. Автомобиль выезжает с парковки в 12:30:01, выезд предоставляется

Сценарий 3:

1. Автомобиль въезжает на парковку в 20:40:06

2. Оплата производится в 22:00:03, заряд = 100 руб.

3. Автомобиль пытается выехать с парковки в 22:30:05, выезд запрещен (прошло более 15 минут с 22:00:03)

4. Дополнительная оплата производится в 22:33:10, дополнительная оплата за период 22: 00: 03-22: 33: 10 = 50 руб. (Общая плата за сеанс = 150 руб.)

5. Автомобиль выходит из парковки в 22:40:00, выезд предоставляется

В расширенной задаче вы также внедрите одно усовершенствование системы, которое позволит регистрировать учетные записи пользователей и выполнять автоматические платежи (возможно, с подключенных кредитных карт или бонусных карт) без необходимости вставлять парковочные талоны в платежный киоск.

В этом случае автоматически свяжет номер автомобиля с системой распознавания изображений на выходе и свяжет его с активным парковочным сеансом (более подробное описание приведено в соответствующем разделе задания).

**3 Описание шаблона проекта**

*Завершите свою работу в прилагаемом консольном приложении, не создавайте новый проект!*

Не меняйте имена классов, методов или свойств, которые вам уже даны. Вы можете выбрать имена новых элементов программы, которые вам будет предложено добавить.

Шаблон содержит определения классов, образующих модель предметной области (ParkingSession, Tariff, User) и центральный класс, который нужно реализовать - ParkingManager.

**4** **Описание задания**

**Базовое задание - 5 баллов**

Для выполнения основной задачи вам не понадобится класс User.

• Изучите два класса - ParkingSession и Tariff - с их свойствами, указанными в шаблоне. Суффикс «Dt» в нескольких свойствах является одним из распространенных способов указать, что это свойство является DateTime. Прежде чем двигаться дальше, поймите логическую связь этих классов с описанием предметной области выше. Найдите объяснение того, почему некоторые свойства в классе ParkingSession могут иметь значение null, а другие - нет.

Примечания:

- Свойство PaymentDt должно хранить дату и время последнего платежа (см. Сценарий 3 в предыдущем разделе)

- Свойство TotalPayment должно хранить общую сумму всех платежей за парковку

• Объявите следующие коллекции как частные поля внутри класса ParkingManager:

- Список активных сессий парковки

- Список завершенных сессий парковки

- Тарифная таблица. Это должно представлять собой таблицу такой же структуры, как указано во введении (вы можете заполнить таблицу своими номерами)3

• Объявите два дополнительных поля / свойства внутри класса ParkingManager, которые будут использоваться в алгоритмах: ParkingCapacity (общее количество мест) и FreeLeavePeriod (это можно взять из первой строки таблицы тарифов, как описано выше)

• Реализуйте следующие методы класса ParkingManager:

- EnterParking

- TryLeaveParkingWithTicket - GetRemainingCost

- PayForParking

Логика, которой должен следовать каждый метод, указана в комментариях внутри кода.

При реализации алгоритмов вы можете обнаружить, что некоторые дополнительные структуры данных могут быть полезны. Если вы используете такие структуры данных надлежащим образом, это даст вам бонус.

• Напишите в основной программе любую логику, которая проверит работу описанных выше методов. Вы можете предоставить свой код с комментариями, которые объясняют, как именно вы проверяете работу класса ParkingManager и его методов в различных сценариях, особенно если это связано с использованием режима отладки.

*Дополнительное задание 1: постоянство данных - 3 балла*

Добавьте методы в класс ParkingManager, чтобы сохранить весь набор данных (сеансы, тарифы, объем парковки) приложения в файле и снова загрузить его при запуске. Вы можете выбрать точный подход (ручной ввод / вывод файла / сериализация), количество файлов (один файл для всего или отдельные файлы для каждого набора данных) и формат файла (текстовый, двоичный).

Обратите внимание, что файл должен сохраняться при каждом изменении данных приложения. Расширьте логику тестирования из базовой части, если это необходимо.

*Дополнительное задание 2: простая система лояльности - 2 балла*

Добавьте следующие функции в ваш класс ParkingManager:

• Добавить коллекцию зарегистрированных посетителей (пользователей). Это также должно быть загружено из файла. Процесс регистрации не является частью этого назначения, вы можете просто предопределить список в файле (однако, он не должен быть жестко задан)

• Добавьте следующую логику в метод «EnterParking»: при создании нового сеанса парковки проверьте, есть ли зарегистрированный пользователь с тем же номером автомобильного номера. Если такой пользователь существует, свяжите сессию парковки с объектом пользователя (вам необходимо настроить класс ParkingSession). При сохранении данных убедитесь, что связь между сеансом и пользователем также сохранена.

• Реализуйте метод TryLeaveParkingByCarPlateNumber, подробное описание которого приведено в комментариях метода в шаблоне.

• Расширьте логику тестирования из базовой части, если это необходимо.

**5 Представление**

Отправьте свое решение, выполнив следующие действия:

1. Удалите две временные папки - «bin» и «obj» из папки проекта. 2. Добавьте всю папку решения в архив ZIP (не RAR или 7Z).

3. Загрузите архив в Canvas LMS в соответствующем разделе

**6 Оценочная политика**

В целом, оценка определяется количеством и качеством выполненных заданий с учетом максимально возможных оценок для каждой из трех частей. Общая оценка может быть снижена в следующих случаях:

• Неэффективная реализация алгоритма (-1 балл)

• Информация дублируется несколько раз в файле (-1 балл) • Плохой стиль программирования (-1 балл)

5