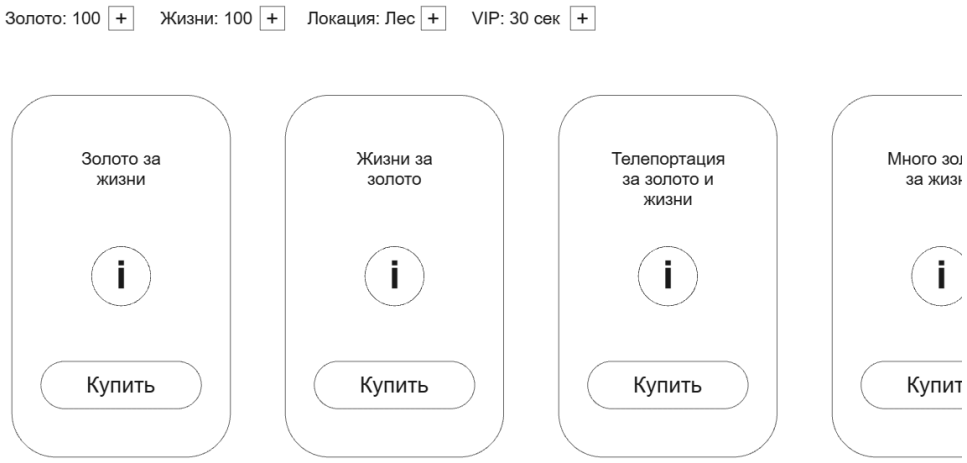
Реализовать в Unity рабочий вариант магазина с изоляцией доменов.

Особое внимание на код и его структуру. Организация ассетов в Unity - вторична. Внешний вид на усмотрение исполнителя.

Требуется сделать простой экран с витриной игрового “магазина”:

- Игрок видит список доступных для покупки бандлов (это одна Unity сцена).



- Карточка каждого бандла содержит название бандла, круглую кнопку “i” и кнопку “Купить”.
- Вверху плашки с описание ресурсов игрока: количество золота, жизней, текущая локация, и длительность VIP статуса (выводим всегда в секундах, обновлять в реальном времени не надо).
- Если у игрока не хватает ресурсов на покупку, то кнопка “Купить” неактивна.
- При покупке симулируется запрос на бекенд в течении 3ех секунд, на это время кнопка “Купить” не активна и на ней показывается надпись “Обработка…”, только после этого происходит реальная покупка и награда.
- При клике на “i” открывается новая Unity сцена, где на полный экран отображается только карточка, на которой была нажата кнопка “i”.
    - Карточка на этой сцене точно такая же, как в списке бандлов, но без кнопки “i”
    - Карточка точно так же полнофункциональная, т.е. можно нажать на купить и т.д.
    - На экране есть кнопка “назад”, чтобы вернуться на сцену со списком бандлов
- Должна быть возможность настраивать бандлы силами геймдизайнеров через инспектор в Юнити (например, ScriptableObject). Не думаем о том, что

награду и траты надо проводить через бекенд, вся логика локальная. Обращение к бекенду мы только симулируем для целей тестового задания.
- Стоимость и награда за бандл должны настраиваться гибко, путем комбинирования “кирпичиков”, которые предоставили программисты для геймдизайнеров. Например, бандл может потратить 10 жизней и 100 золота, но в замен перенести в новую локацию и дать временный VIP статус. Количество “кирпичиков” в стоимости и награде может быть любым.
    - Обратите внимание: это НЕ магазин за реальные деньги.
- Стоимость и награда за бандл на карточке не видны, чтобы не усложнять задачу.
- Мы хотим, чтобы у нас была упрощенная доменная модель, когда в коде есть множество доменов (в виде отдельных папок с asmdef) и они ничего друг о друге не знают, а общее знание имеют только через один домен “ядро”. Например, есть домен “здоровья”, где ведется вся бизнес логика, связанная с жизнями. Домен “локации”, где обрабатывается поведения на какой карте находится игрок. Домен “VIP”, где ведется временный VIP статус. И другие.
- При этом “магазин” с бандлами - это тоже домен, который не должен ничего знать о других параллельных доменах.
- Мы хотим, чтобы соблюдался принцип модульности, т.е. добавление новых “фичей” в игру не должно постоянно увеличивать код какого-либо домена, т.е. не должно существовать God-домена. То есть если мы захотим добавить новый домен, например, “лидерборды” и при этом иметь возможность продавать “очки” лидерборда в магазине, то нового кода в домене “магазина” не должно появиться, все нововведения только в рамках домена “лидерборды”. “Ядро” может иногда дополняться, но появление нового домена, обычно, не повод для этого.
    - Подсказка: в ядре не должно быть классов с перечислением сущностей или понятий из других доменов.
- Мы договариваемся, что появление нового “кирпичика” для настройки награды и стоимости, происходит внутри каждого домена отдельно. Т.е. если за покупку бандла должны истратиться жизни, то “кирпичик” траты жизней должен лежать в домене (папке) “здоровья”. А если в награду мы хотим переместить игрока в другую локацию, то “кирпичик” награды, перемещающий в локацию, должен лежать в домене (папке) “локаций”.

Технические требования:
- Есть 6 доменов:

    - Core - ядро, сюда можно положить общие базовые понятия, которые могут понадобиться всем доменам, например, базовый тип ScriptableObject, который вводит возможность проверить доступно ли действие и если доступно, то его применить, либо какой-то общий interface.
        - Подсказка. Здесь нужны одна-две общих сущности, но при этом они не должны в себе иметь отсылки и другим доменам.
        - Дополнительно Core должен содержать singleton PlayerData, к которому есть доступ у других доменов. Подробнее в следующих пунктах.
    - Health - домен, отвечающий за здоровье.
    - Gold - домен, отвечающий за золото.
    - Location - домен, отвечающий за текущую локацию игрока. Пусть локация будет просто текстовым значением.
        - Подсказка. Обратите внимание, что тратиться и вручаться могут не только численные значения, но и текст, и время, и все это в любой комбинации в зависимости от потребностей домена, т.е. “кирпичики” для “бандлов” не могут быть описаны через единую структуру.
    - VIP - домен, отвечающий за VIP-статус. VIP-статус - это просто время на которое доступен VIP, хранится в TimeSpan.
    - Shop - домен, отвечающий за магазин.
- Домены - это папки-ассембли (с вложенными .asmdef), при этом ассембли Health, Gold, Location, VIP и Shop имеют зависимости от Core, и не имеют зависимостей друг на друга.
- Весь код проекта должен находиться строго в этих доменах, никакие другие домены и папки с кодом не допускаются.
- Внутри папок-доменов код организован так, что один класс или интерфейс - это один файл.
- Для упрощения разрешается использовать паттерн Singleton, так как игрок всегда один.
- Ассортимент магазина должен настраиваться гибко через ScriptableObject или другим способом, но обязательно в Unity Editor. Магазин состоит из бандлов, где у каждого бандла может быть какая угодно цена и какая угодно награда, обе в любой комбинации и количестве.
- Должны быть реализованы следующие “кубики” для трат и наград:
    - дать/забрать фиксированное кол-во золота;
    - дать/забрать фиксированное кол-во здоровья;
    - дать/забрать процентное кол-во здоровья от текущего;
    - изменить текущую локации на заданную (текстом);

    - дать/забрать фиксированное время VIP статуса.
- Нужен несложный GUI, который выводит в ряд карточки бандлов, где каждая карточка это: большая надпись с названием бандла и кнопка “купить”, причем кнопка купить неактивна, если у игрока не хватает ресурсов. Выводить цену или награду - не надо.
- Когда игрок нажимает “купить”, то симулируется сетевой запрос на 3 секунды, на это время на кнопке “купить” надпись заменяется на “Обработка…”.
- Вверху в GUI выводится:
    - текущее количество золота
    - текущее количество здоровья
    - текущая локация
    - текущий VIP-статус

  без графических изысков. Ориентир на MVC подход.
- Бизнес логика по каждой сущности из пункта выше должна быть реализована строго в своем домене. Т.е. каждый домен полностью изолирован. Это означает, что в Core не должно быть ничего, что могло бы прирастать кодом при появлении новых доменов. А также не должно быть какого-то очень высокого-уровневого домена по типу GUI.
    - Подсказка: ориентируемся на базовый MVC подход, когда нужные контроллеры реализованы внутри соответствующих доменов.
- При этом обязательно в Core надо реализовать singleton PlayerData, к которому могут обращаться все домены. В PlayerData надо хранить все “перманентные” данные игрока, например, домен здоровья должен хранить в нем здоровье, домен локации - текущую локацию. При этом обязательно! В PlayerData не могут фигурировать явные поля из знания конкретных доменов, т.е. не может быть поля health, location, vipDuration и т.д.
    - Не требуется реализовывать сохранение на диск, достаточно простого рантайм хранилища.
    - Подсказка: подумайте об абстракциях и generic подходе.
- В GUI рядом с каждым выведенным текущим количеством ресурса из пунктов выше (текущее здоровье, текущее золото, текущая локация) должна быть кнопка “+”, которая по типу чита увеличивает текущее количество соответствующего ресурса, а локация сбрасывается в дефолтную. При этом важно, чтобы кнопки “купить” у соответствующих бандлов стали активными, если новое количество ресурса стало достаточным для покупки.