ПЕРЕД НАЧАЛОМ ПРОЧТЕНИЯ ДОКУМЕНТАЦИИ НАСТОЙЧИВО РЕКОМЕНДУЮ ОЗНАКОМИТЬСЯ С РЕАКТИВНЫМ ПОДХОДОМ К НАПИСАНИЮ КОДА

1. **Reactive**

**Example:**

Reactive<variable> reactive = new Reactive<variable>();

ReactiveList<variable> reactiveList = new ReactiveList<variable>();

**Fields:**

* value(get; set;) – значение переменной

**Mhetods:**

* void ConnectToSaver(KEY) – активирует автоматическое сохранение перменной по ключу

KEY – ключ для сохранения

* IDisposable Subscribe<T>(Action<T>) – подписка на изменение переменной, ACTION вызываеться при измененении значения value
* IDisposable SubscribeAndInvoke<T>(Action<T>) – подписка на изменение переменной, ACTION вызываеться сразу, и далее при измененении значения value
* void UnsubscribeAll – отписывает
* IDisposable SubscribeForEach(Action<T, CollectionEvent> onChangeElement) (for ReactiveList) вызов при удалени, добавлении или замене конкретного обьекта в списке

1. **Saves**

**Desctiption:**

Сохранения работают на основе парадигмы реактивного программирования, применяеться к каждой отдельной переменной или списку.

Reactive<variable> reactive = new Reactive<variable>();

ReactiveList<variable> reactiveList = new ReactiveList<variable>();

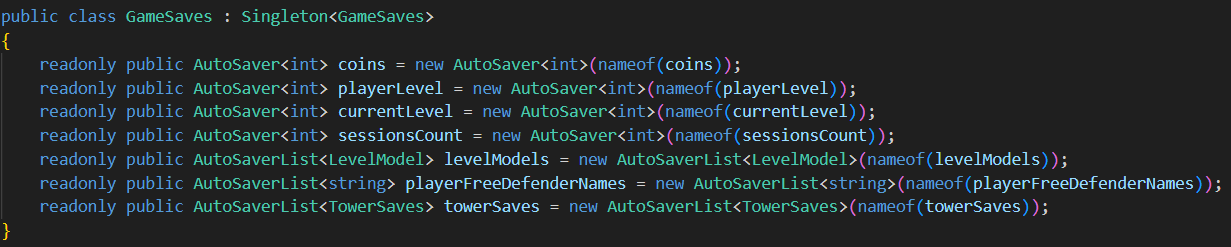
Таким образом инициализируеться реактивная перемення или колекция, чтобы подключить переменную к сохранениям, необходимо вызвать метода ConnectToSaver(KEY) где в качестве ключа выступает любой стринг, по этому ключу метод найдет последнее сохранение. Далее при любом измененении переменной, или взаимодецствии со списком, автоматически будет вызываться пересохранене данных. Таким образом вы избавитесь от нгужды постоянно думать о сохраненях в своей игре, вам необходимо будет лишь создать реактивную переменную или список, подключить его к хранителю данных по ключу, ипри любом изменении значений, данные будут пересохранены мгновенно.

**Example:**



Или использоваь класс AutoSaver/AutoSaverList как упрощение.

**Example:**



**Connections**

**Desctiption:** используеться для отмены подписки на реактивное событие

**Mhetods:**

DisconnectAll() – после вызова метода, все подключенные подписки отпишуться

**Example:**

Connections connections = new();

connections += reactive.Subscribe(value => value);

connections.DisconnectAll();//все подписки удаляються, и строка выше не являеться действительной

1. **PlayerPrefsPro**

**Desctiption:** аналог PlayerPrefs в Unity, но способен сохранять любые сериализуемые обьекты

**Mhetods:**

void Set(key, object) –сохраняет любой сериализуемый обьект по ключу

T Get<T>(key) – возвращает обьект по ключу приводя его к T

void DeleteKey(key) – удаляет сохранение по ключу

void DeleteAllKeys() – очищает все сохранения

void SetBytes(this byte[] bytes, string key) – сохраняет массив байтов по ключу

byte[] GetBytes(string key) – возвращает сохраненные байты по ключу

string GetPatch(string key) – возвращает путь к файлу по ключу

Sprite GetSprite(string key) – возвращает сохраненный спрайт

void SetSprite(Sprite sprite, string key) – сохраняет спрайт по ключу

List<string> GetAllKeys() – возвращает все существующие ключи по которым есть сохранения

bool HasKey(string key) – существует ли сохранение по ключу

1. **Window Managment**

**Desctiption:**

Для использования необходимо обратиться к WindowManager.Instance, префаб канваса на сцене создаеться автоматически.

В методе Show<Window>() необходимо указать тип окна которое вы хотите открыть/создать по указанному типу окно будет найдено в папке ресурсов, и заспавнено на канвас

Для создания окна необходимо унаследовать класс окна от WindowBase или AnimatedWindowBase после чего создать префаб окна на канвасе, и перетащить его в папку Resources вашего проекта

**WindowManager**

**Fields:**

WindowManager Instance – статический инстанс

Canvas canvas – возвращает канвас менеджера окон

**Mhetods:**

Window Show<Window>() - открыть/создать окно на канвасе по указанному типу, вернет созданный екземпляр

void CloseTop() – закрыть последнее открытое окно

void Close(Window) – закрыть указанное окно

void CloseAll() – закрыть все открытые окна

T GetOpendWindow<T>() – получить последнее открытое окно по указанному типу

**WindowBase/AnimatedWindowBase**

**Fields:**

UnityEvent OnClose – событие закрытия окна

bool isReusableView – если true то после закрытия префаб не удалиться и будет переиспользован при следующем открытии

Button closeButton – кнопка закрівающая окно, может біть null

Connections connections – для взаимодействия с реактивными переменными и UI, при закрытии окна автоматически очищает все привязанные ивенты

**Mhetods:**

* void Close() - закрыть окно
* virtual void OnOpend() – вызываеться при открытии окна