Technická dokumentace
Universal Inventory System

Univerzální knihovna pro správu inventáře ve videohrách, navržená pro integraci do různých herních enginů, například Unity. Knihovna poskytuje vývojářům širokou škálu funkcí pro práci s inventářem, včetně podpory různých typů položek, omezení kontejnerů, serializace dat, třídění, filtrování a vyhledávání položek. Je navržena tak, aby byla flexibilní a rozšiřitelná.

Hlavní cíl a úkoly

Hlavním cílem systému Universal Inventory System je vytvoření univerzálního řešení pro správu zásob, které lze snadno integrovat do jakéhokoli projektu vyžadujícího ukládání a manipulaci se soubory položek.

Mezi hlavní úkoly projektu patří:

Integrace do herních enginů: Zajištění snadné integrace knihovny do populárních herních enginů, jako je Unity, pro správu inventáře.

Podpora různých typů položek: Knihovna umožňuje pracovat s různými typy položek, včetně položek s různými vlastnostmi, jako je hmotnost nebo typ.

Omezení pro kontejnery: V případě, že se jedná o položky, které jsou v kontejneru, je možné je použít pro všechny položky: Ke kontejnerům můžete snadno přidávat omezení, která řídí počet, hmotnost a typy přidávaných položek.

Třídění a filtrování položek: Vestavěné metody umožňují třídit položky podle libovolných vlastností a filtrovat je podle zadaných kritérií.

Serializace do JSON a XML: Knihovna podporuje serializaci a deserializaci inventáře ve formátech JSON a XML, což umožňuje ukládat a obnovovat stav inventáře.

Lokalizace: Knihovna podporuje možnost lokalizovat vlastnosti položek, což usnadňuje její začlenění do vícejazyčných projektů.

Modularita: Knihovna je postavena na bázi rozhraní a abstraktních tříd, což usnadňuje rozšíření její funkčnosti nebo její přizpůsobení specifickým potřebám hry.

Architektura projektu

Celková struktura projektu

Projekt Univerzální inventární systém se skládá ze tří částí:

1. UniversalInventorySystemLibrary:

Jedná se o jádro projektu, které obsahuje hlavní funkce inventárního systému. Je rozdělena do několika částí:

Items: Obsahuje rozhraní a třídy pro reprezentaci položek.

Serialiser: Obsahuje serializátory pro ukládání a obnovu stavu inventáře.

Attributes: Definuje atributy pro metadata vlastností položek.

Limiters: Obsahuje omezovače pro kontrolu položek v kontejnerech.

Container: Slouží k omezení limitů: Implementuje základní operace s kontejnery položek.

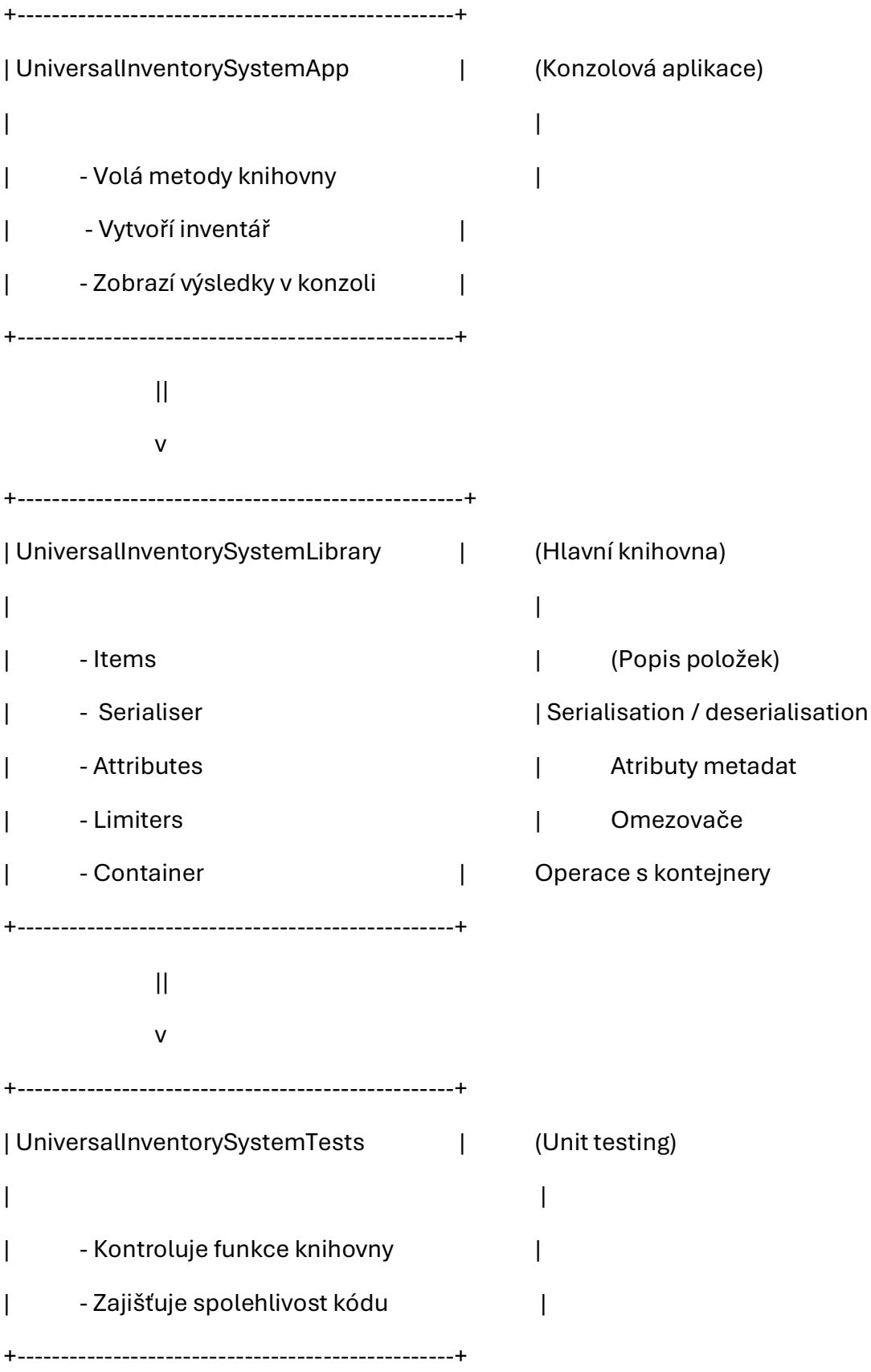
2. UniversalInventorySystemApp:

Jedná se o konzolovou aplikaci, která demonstruje použití knihovny UniversalInventorySystemLibrary. Vytváří inventář, přidává položky, třídí je a filtruje a zobrazuje výsledky v konzoli.

3. UniversalInventorySystemTests:

Obsahuje testy pro kontrolu správné činnosti součástí knihovny.

Propojení součástí: Schéma



Popis komponent knihovny

Sekce Items:

Část Items (Položky) knihovny univerzálního inventárního systému je zodpovědná za definici a správu položek, které lze přidat do inventáře. Tato sekce se skládá z několika klíčových rozhraní a tříd, které zajišťují flexibilitu a rozšiřitelnost systému.

1. Rozhraní objektů

1. 1. Rozhraní IItem

Rozhraní IItem je základní rozhraní pro všechny položky v inventárním systému. Slouží jako základ pro vytváření různých typů položek, které lze přidávat do inventury.

1.2 IItemWithWeight

Rozhraní *IItemWithWeight* rozšiřuje základní rozhraní *IItem* a přidává vlastnost pro uložení hmotnosti položky.

Vlastnost *Weight*: Definuje hmotnost položky, která je důležitým parametrem pro omezení kontejneru, jako je maximální přípustná hmotnost.

1.3 IItemWithType<EnumT> where EnumT: Enum

Rozhraní *IItemWithType<EnumT>* rozšiřuje základní rozhraní *IItem*, přidává typ položky definovaný výčtem *EnumT*.

Vlastnost *ItemType*: Představuje typ položky, který lze použít ke kategorizaci nebo omezení položek v kontejneru na základě typu.

2. Class *BaseItem*

Třída BaseItem implementuje rozhraní IItem a slouží jako základní třída pro položky v inventárním systému. Poskytuje vlastnost Name pro uložení názvu položky.

Pole *_name*: Soukromé pole pro uložení názvu položky.

Vlastnost *Name*: Veřejná vlastnost, která vrací hodnotu *_name*. Je označena atributem *ItemProperty*, který označuje, že tato vlastnost nemá být při zpracování ignorována.

Konstruktor *BaseItem*(*string name*): Inicializuje novou položku se zadaným názvem.

3. Class ItemUtils

Třída *ItemUtils* poskytuje utilitární metody pro práci s položkami. Tyto metody usnadňují získávání informací o vlastnostech položek, jako je název, klíč lokalizace a hodnoty vlastností.

3.1 Class ItemPropertyInfo

Interní třída *ItemPropertyInfo* slouží k reprezentaci informací o vlastnosti položky.

Vlastnost *Name*: Název vlastnosti objektu.

NameLocationKey vlastnost: Klíč lokalizace pro jméno vlastnosti, který se používá k načtení lokalizovaného textu.

Value vlastnosti: Hodnota vlastnosti.

3.2 Metoda GetItemPropertiesInfo

Metoda *GetItemPropertiesInfo* získá informace o všech vlastnostech položky, které nejsou označeny jako ignorované, a vrátí je jako seznam objektů *ItemPropertyInfo*.

Parametr *item*: Instance položky, pro kterou chcete získat informace.

Vrací: Seznam objektů *ItemPropertyInfo*, které obsahují informace o vlastnostech položky, včetně jejich názvů, lokalizačních klíčů a hodnot.

Sekce Serializer:

Sekce Serializer knihovny Universal Inventory System je zodpovědná za serializaci a deserializaci položek v různých formátech, například JSON a XML. Serializace umožňuje ukládat stav položek do souborů nebo je přenášet po síti a deserializace umožňuje tento stav obnovit.

1. Rozhraní ISerializer

Rozhraní *ISerializer* definuje smlouvu pro třídy serializátorů. Deklaruje metody pro serializaci a deserializaci objektů.

Metoda *Serialize*: Přijímá objekt a vrací jeho serializovanou reprezentaci jako řetězec.

Metoda *SerialiseAsync*: Asynchronní verze metody *Serialise*, která provádí serializaci na druhém plánu.

Metoda *Deserialize<T>*: Přijímá serializovaný řetězec a vrací obnovený objekt typu *T*.

Metoda *DeserializeAsync<T>*: Asynchronní verze metody *Deserialise*, která provádí deserializaci na druhém plánu.

2. Class JsonSerializer

Třída *JsonSerializer* implementuje rozhraní *ISerializer* a je zodpovědná za serializaci a deserializaci objektů ve formátu JSON.

Metoda *Serialise*: Používá knihovnu *JsonConvert* k převodu objektu do formátu JSON.

Metoda *SerialiseAsync*: Provádí serializaci asynchronně pomocí metody *Task.Run*, která umožňuje, aby serializace probíhala na pozadí.

Metoda *Deserialise<T>:* Převede řetězec JSON zpět na objekt typu *T*.

Metoda *DeserialiseAsync<T>*: Provádí deserializaci asynchronně.

3. Class XMLSerializer

Třída *XMLSerializer* implementuje rozhraní *ISerializer*, provádí serializaci a deserializaci ve formátu XML.

Metoda *Serialize*: Používá třídu *XmlSerializer* k převodu objektu do formátu XML. Objekt je zapsán do řetězce pomocí *StringWriter*.

Metoda *SerialiseAsync*: Asynchronně provede serializaci do XML stejným přístupem jako metoda *Serialise*.

Metoda *Deserialize<T>*: Převede řetězec XML zpět na objekt typu *T* pomocí třídy *XmlSerializer*.

Metoda *DeserializeAsync<T>*: Asynchronně provede deserializaci do formátu XML.

Secke Attributes:

Část Attributes knihovny Universal Inventory System slouží k vytváření vlastních atributů, které rozšiřují funkčnost položek v inventáři. Tyto atributy slouží k označení vlastností položek a poskytují další informace, které lze použít během aplikace, například pro lokalizaci nebo řízení zobrazení vlastností.

1. Atribut ItemProperty

Atribut *ItemProperty* slouží k označení vlastností položky, které jsou důležité pro zpracování nebo zobrazení v inventárním systému. Umožňuje označit vlastnosti, které lze ignorovat, nebo naopak by měly být při práci s položkami brány v úvahu.

Vlastnost *Ignore*: Označuje, zda má být vlastnost ignorována. Pokud je hodnota *Ignore* nastavena na *true*, bude vlastnost při zpracování, například při serializaci nebo zobrazení, ignorována.

Konstruktor *ItemProperty(bool ignore)*: Umožňuje nastavit hodnotu vlastnosti *Ignore* při použití atributu na konkrétní vlastnost položky.

2. Atribut PropertyLocalisationKey

Atribut *PropertyLocalisationKey* slouží k definování lokalizačních klíčů pro vlastnosti objektu. Umožňuje svázat vlastnost s konkrétním klíčem v souboru zdrojů, který obsahuje přeložené hodnoty pro různé jazyky.

Vlastnost *Key*: Uloží lokalizační klíč, který bude použit pro vyhledání přeložené hodnoty v souborech zdrojů.

Konstruktor *PropertyLocalizationKey(string key)*: Přijímá řetězec klíče, který odpovídá lokalizačnímu klíči pro konkrétní vlastnost.

Atribut *[AttributeUsage]*: Určuje, že atribut *PropertyLocalisationKey* lze použít pouze pro pole nebo vlastnosti, není dědičný a lze jej použít pouze jednou pro jednu vlastnost nebo pole.

Sekce Limiters:

Sekce Limiters knihovny univerzálního inventárního systému je zodpovědná za omezení počtu nebo vlastností položek, které lze přidat do kontejneru. Omezovače umožňují řídit, kolik položek určitého typu, hmotnosti nebo celkového množství lze do zásobníku umístit.

1. Rozhraní IContainerLimiter

Rozhraní *IContainerLimiter* definuje smlouvu pro omezovače kontejnerů. Deklaruje metody, které kontrolují, zda lze do kontejneru přidat jednu nebo více položek.

Metoda *CanAddItem*: Zkontroluje, zda lze do kontejneru přidat jednu položku s danými omezeními.

Metoda *CanAddItemArray*: Zkontroluje, zda lze do kontejneru přidat více položek. K rozdělení položek na ty, které lze přidat, a ty, které přidat nelze, se používají dva seznamy (*canAddItems* a *cannotAddItems*).

2. Class ContainerLimiter

ContainerLimiter je základní třída pro všechny omezovače kontejnerů. Implementuje rozhraní *IContainerLimiter* a poskytuje základní funkce pro správu kontejneru.

Pole *_itemContainer*: Chráněné pole, které obsahuje odkaz na kontejner položky, na který se vztahují omezení.

Metoda *CanAddItem*: Abstraktní metoda, která musí být implementována v odvozených třídách. Kontroluje, zda je možné přidat do kontejneru jednu položku.

Metoda *CanAddItemArray*: Abstraktní metoda, která musí být rovněž implementována v odvozených třídách. Kontroluje, zda lze přidat více položek.

3. Class CapacityLimiter

Třída *CapacityLimiter* je specifickou implementací třídy *ContainerLimiter*, která omezuje počet položek v kontejneru. Tento omezovač zajišťuje, aby počet položek v kontejneru nepřekročil zadanou hodnotu.

Pole *_capacity*: Ukládá maximální počet položek, které lze do kontejneru umístit.

Metoda *CanAddItem*: Kontroluje, zda je v kontejneru dostatek místa pro přidání jedné položky.

Metoda *CanAddItemArray*: Zkontroluje, zda lze přidat více položek, a rozdělí je na ty, které lze přidat, a ty, které nelze přidat kvůli nedostatku místa.

4. Class TypeLimiter<EnumT>

Třída *TypeLimiter<EnumT>* je omezovač, který řídí počet položek určitého typu v kontejneru. Typy položek jsou definovány pomocí výčtu *EnumT*.

Pole *_limits*: Slovník, který uchovává maximální počet položek pro každý typ.

Konstruktor *TypeLimiter*: Konstruktor přebírá kontejner položek (*IItemContainer*) a slovník limitů (*Dictionary<EnumT*, int>). Zavolá základní konstruktor *ContainerLimiter* a inicializuje pole *_limits*.

Metoda *CanAddItem*: Tato metoda zjišťuje, zda lze do kontejneru přidat jednu položku. Pokud typ položky odpovídá jednomu z typů ve slovníku *_limits* a počet položek tohoto typu v kontejneru je menší než nastavený limit, metoda vrátí *true*, jinak *false*.

Metoda *CanAddItemArray*: Tato metoda zjišťuje, zda je možné přidat několik položek najednou. Nejprve spočítá aktuální počet položek každého typu a určí, zda lze přidat všechny položky v poli položek v rámci nastaveného limitu. Položky, které lze přidat, se přidají do seznamu *canAddItems* a ty, které překročí limit, se přidají do seznamu *cannotAddItems*.

Metoda *GetTotalCountItemWithType*: Tato metoda spočítá celkový počet položek určitého typu v kontejneru.

Metoda *GetTotalCountItemWithTypes*: Tato metoda aktualizuje slovník s počtem položek jednotlivých typů v kontejneru. Slovník je předán jako parametr a počet položek v něm je aktualizován pro každý typ.

Metoda *ToString*: Přepsaná metoda *ToString*, která vrací řetězcovou reprezentaci třídy *TypeLimiter<EnumT>* s informacemi o limitech nastavených pro jednotlivé typy položek.

5. Class WeightLimiter

Třída *WeightLimiter* omezuje celkovou hmotnost položek v kontejneru.

Pole *_maxWeight*: Maximální přípustná hmotnost položek v kontejneru.

Metoda *CanAddItem*: Kontroluje, zda celková hmotnost položek v kontejneru po přidání nové položky nepřekročí maximální hmotnostní limit.

Metoda *CanAddItemArray*: Kontroluje, zda lze přidat více položek, aniž by byl překročen hmotnostní limit. Pokud hmotnost některé položky překročí limit, je přidána do seznamu *cannotAddItems*.

Sekce Container

Sekce Container je zodpovědná za správu kontejnerů pro položky. Kontejner je objekt, který uchovává položky (například inventář ve hře). Umožňuje přidávání, třídění, filtrování a vyhledávání položek a také poskytuje prostředky pro zobrazení informací o těchto položkách.

1. Class BaseItemContainer

Tato třída je základní implementací kontejneru položek. Je zodpovědná za ukládání položek a základní operace s nimi, jako je přidávání a načítání seznamu všech položek.

Pole *_items*: Jedná se o soukromé pole, které uchovává seznam položek v kontejneru.

Metoda *Add*: Přidá do kontejneru jednu položku.

Metoda *AddRange*: Přidá do kontejneru více položek.

Metoda *GetItems*: Vrací seznam všech položek obsažených v kontejneru.

Vlastnost *Count*: Vrací počet položek v kontejneru.

Vlastnost *IsReadOnly* vždy vrací *false*, což znamená, že kontejner podporuje přidávání a odebírání položek, tj. není určen pouze pro čtení.

Metoda *Clear* vyčistí kontejner tak, že odstraní všechny položky ze seznamu položek.

Metoda *Contains*: zjišťuje, zda se v kontejneru nachází určitá položka. Vrací *true*, pokud je položka přítomna, a *false* v opačném případě.

Metoda *CopyTo*: zkopíruje položky z kontejneru do pole začínajícího na zadaném indexu.

Metoda *GetEnumerator*: vrací iterátor pro iteraci všech položek v kontejneru.

Metoda *Remove*: odstraní konkrétní položku z kontejneru a vrátí *true*, pokud byla položka úspěšně odstraněna, a *false*, pokud nebyla v kontejneru nalezena.

Metoda *ToString*: vrací řetězcovou reprezentaci všech položek v kontejneru, včetně jejich typů a vlastností.

2. Class ItemContainerConsoleView

V současné době ještě není třída ItemContainerConsoleView implementována, ale její potenciální účel spočívá v tom, že může být použita k zobrazení informací o obsahu kontejneru na konzole.

3. Class Filterer

Třída Filterer je zodpovědná za filtrování položek v kontejneru na základě určitých kritérií. Filtrování je proces výběru pouze těch položek, které splňují určité podmínky.

enum ComparisonType: Tento výčet definuje typy porovnání, které lze provádět při filtrování:

- Porovnává se na rovnost.
- GreaterThan - Kontroluje, zda je hodnota větší než zadaná hodnota.
- LessThan - Testuje, zda je hodnota menší než zadaná hodnota.

Metoda *CompareValues*: Tato soukromá metoda provádí porovnání hodnot na základě typu porovnání předaného v parametrech. Používá výchozí porovnávací funkci pro typ ш a vrací *true*, pokud je porovnání úspěšné, a *false* v opačném případě.

Metoda *Filter*: Tato metoda filtruje seznam položek na základě určité vlastnosti *fieldName* a hodnoty *value*. Kontroluje každou položku v seznamu a přidá ji do výsledků, pokud hodnota jejího pole odpovídá kritériím porovnání.

Metoda *FilterAsync*: Jedná se o asynchronní verzi metody *Filter*, která provádí filtrování v asynchronním režimu. Je užitečná pro velké seznamy, kde se chcete vyhnout blokování hlavního vlákna během filtrování.

4. Class Finder

Třída Finder je zodpovědná za vyhledávání položek v kontejneru. Vyhledávání je proces hledání konkrétních položek na základě určitých kritérií (například položka s konkrétním názvem nebo hodnotou vlastnosti).

Metoda *Find*: Vyhledá položky, které mají určitou vlastnost (*fieldNamel*) s danou hodnotou (*value*). Vrátí seznam položek, které těmto kritériím odpovídají.

Metoda FindAsync: Jedná se o asynchronní verzi metody *Find*, která provádí vyhledávání v asynchronním režimu. Umožňuje provádět dlouhé vyhledávací operace v samostatném vlákně bez blokování hlavního vlákna aplikace.

Hlavní třída Inventory

Třída Inventory představuje inventární systém, který může obsahovat a spravovat položky. Tato třída poskytuje různé metody pro přidávání, vyhledávání, třídění a filtrování položek a také pro serializaci a deserializaci stavu inventáře. Inventář používá kontejner pro ukládání položek, omezovače pro řízení přidávání položek a serializátory pro převod inventáře do a z formátu řádku.

Vlastnost *ItemsCount*: Vrací počet položek v inventáři.

Vlastnost *Items*: Vrací seznam položek obsažených v inventáři.

Metoda *TryAdd(IItem item)*: Pokus o přidání položky do inventáře. Vrací true, pokud byl předmět úspěšně přidán, jinak false. Zkontroluje, zda lze položku přidat pomocí kontejneruLimiter, a poté položku přidá do kontejneru.

Metoda *TryAddRange(ICollection<IItem> newItems, List<IItem> canAddItems, List<IItem> cannotAddItems, bool allOnly = true)*: Pokus o přidání více položek do inventáře. Vrací true, pokud byly přidány všechny položky, jinak false. Přidá položky pouze v případě, že lze přidat všechny položky, nebo přidá pouze ty, které lze přidat.

Metoda *Serialise*(): Serializuje inventář do řetězce pomocí zadaného serializéru.

Metoda *SerializeAsync*(): Asynchronně serializuje inventář do řetězce.

Metoda *Deserialise(string value)*: Deserializuje inventář z řetězce a obnoví jeho stav.

Metoda *SortByName*(): Seřadí položky v inventáři podle jejich názvu.

Metoda *Sort<T>*(string *propertyName*) *whereT : class*: Seřadí položky podle zadané vlastnosti.

Metoda *Filter<T>(string fieldName, object value, ComparisonType comparisonType) where T : class*: Filtruje objekty podle zadané vlastnosti a typu porovnání.

Metoda *Find<T>(string fieldName, object value) where T : class*: Najde položky v inventáři na základě zadané vlastnosti a hodnoty.

Struktura testovacího projektu UniversalInventorySystemTests

Globální usings:

global using Microsoft.VisualStudio.TestTools.UnitTesting;

BaseTest:

1. Test přidání jedné položky (*TestAddItem*)

Scénář: Tento test ověřuje, zda je možné přidat jednu položku do inventáře s omezením počtu položek (kapacitní limit).

Popis: Nejprve se vytvoří inventář s omezením počtu položek na 5 pomocí metody CreateInventoryWithCapacityLimiter.

Do inventáře se přidají dvě položky.

Zkontroluje se, zda počet položek v inventáři je 2.

Očekávaný výsledek: Test by měl potvrdit, že dvě přidané položky jsou v inventáři.

2. Test přidání více položek (*TestAddItemRange*)

Scénář: Tento test ověřuje možnost přidat do inventáře více položek najednou.

Popis: Je vytvořen inventář s limitem 5 položek.

Pomocí metody TryAddRange se vytvoří pole dvou položek a přidá se do inventáře.

Zkontroluje se počet položek v inventáři a správnost pořadí přidaných položek.

Očekávaný výsledek: Test by měl potvrdit, že všechny položky byly úspěšně přidány a jsou ve správném pořadí.

3. Test třídění objektů (*TestSort*)

Scénář: Tento test ověřuje, zda můžete třídit předměty v inventáři podle názvu.

Popis: Je vytvořen inventář s limitem 5 položek.

Jsou přidány tři položky s různými názvy.

Položky jsou seřazeny podle názvu pomocí metody SortByName.

Zkontroluje se, zda jsou položky správně seřazeny.

Očekávaný výsledek: Test by měl potvrdit, že položky jsou seřazeny abecedně podle názvu.

4. Testovací filtrování objektů (*TestFilter*)

Scénář: Tento test ověřuje schopnost filtrovat objekty na základě jejich hmotnosti pomocí WeightLimiter.

Popis: Je vytvořen inventář s limitem 4 položek.

Jsou přidány tři položky s různou hmotností.

Filtrování se provádí pro položky s hmotností menší než 5 jednotek.

Kontroluje se, zda výsledek filtru obsahuje pouze ty položky, které splňují dané kritérium.

Očekávaný výsledek: Test by měl potvrdit, že filtrování funguje správně a vrací pouze položky s hmotností menší než 5 jednotek.

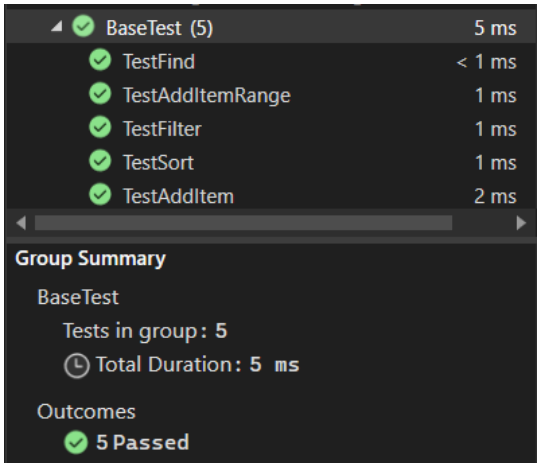
5. Testovací vyhledávání objektu (*TestFind*)

Scénář: zkontrolujte, zda metoda Find ve třídě Inventory správně najde položky v inventáři podle zadané vlastnosti (Name) a hodnoty („item2“).

Popis: Použijeme metodu CreateInventoryWithCapacityLimiter k vytvoření inventáře s kapacitním omezením 5 položek. Vytvoříme dva objekty třídy BaseItem s názvy „item1“ a „item2“. Zavoláme metodu Find a předáme jí název vlastnosti „Name“ a hodnotu „item2“.

Očekávaný výsledek: Zkontrolujte, zda metoda Find vrátila seznam obsahující přesně jednu položku. Zkontrolujte, zda je název této položky „item2“.

Test results



CapacityLimiterTest

1. Test přidávání položek až do dosažení limitu (*TestCapacityLimiterAddItem*)

Scénář: Tento test ověřuje, že po dosažení nastaveného limitu (v tomto případě 5 položek) nelze do inventáře přidat žádné nové položky.

Popis: Nejprve se vytvoří inventář s limitem 5 položek pomocí metody CreateInventoryWithCapacityLimiter.

Poté se do inventáře přidá 5 položek.

Pokus o přidání šesté položky by měl vrátit false.

Očekávaný výsledek: Test by měl potvrdit, že po dosažení limitu nelze do inventáře přidat šestou položku.

2. Test přidání více položek najednou (*TestCapacityLimiterAddRangeItem*)

Scénář: Tento test ověřuje, jak CapacityLimiter zvládá přidání více položek najednou, když některé z nich lze přidat a některé z nich překročí limit.

Popis: Je vytvořen inventář s limitem 5 položek.

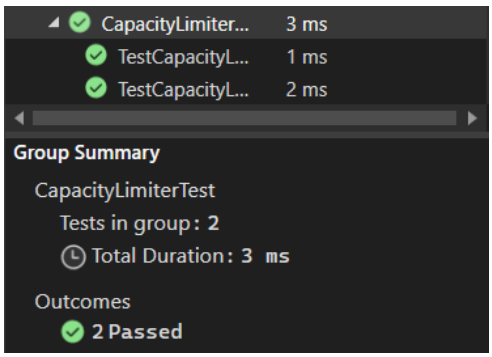
Do inventáře jsou přidány 4 položky.

Je proveden pokus o přidání dvou položek najednou, což vede k překročení limitu.

Seznamy canAdd a cannotAdd musí obsahovat odpovídající položky, které lze a nelze přidat.

Očekávaný výsledek: Test by měl potvrdit, že do soupisu je přidána pouze jedna položka a druhá položka je odmítnuta.

Test Result



TypeLimiterTest

1. Test přidání položek určitého typu (*TestTypeLimiterAddItem*)

Scénář: Tento test ověřuje, zda TypeLimiter správně zablokuje přidávání položek při dosažení limitu pro určitý typ položky.

Popis: Je vytvořen inventář s omezením počtu položek typu Tool (limit: 1) a Plant (limit: 2).

Je přidána jedna položka typu Tool (Nástroj) a je kontrolováno, že nelze přidat druhou položku stejného typu.

Kontroluje se, že položky typu Plant lze přidávat, dokud není dosaženo limitu.

Očekávaný výsledek: Test by měl potvrdit, že položky typu Tool jsou po dosažení limitu blokovány a položky typu Plant lze přidávat, dokud není dosažen jejich limit.

2. Test přidání více položek najednou (*TestTypeLimiterAddRangeItem*)

Scénář: Tento test ověřuje, jak TypeLimiter zvládá přidání více položek najednou, když některé z nich lze přidat a některé z nich překročí limit.

Popis: Je vytvořen inventář s omezením počtu položek typu Tool (limit: 2) a Plant (limit: 2).

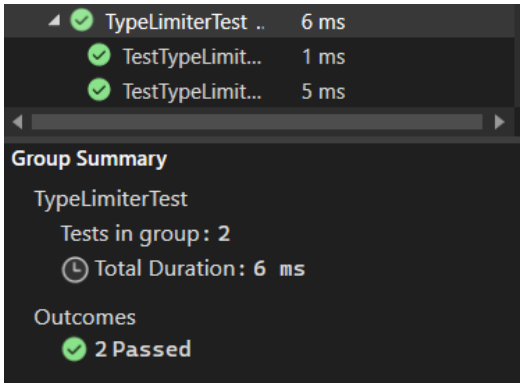
Jsou přidány dvě položky typu Tool (Nástroj) a jedna položka typu Plant (Rostlina).

Je proveden pokus o přidání dvou položek najednou: jedné typu Plant (rostlina) a jedné typu Tool (nástroj).

Kontroluje se, zda je do inventáře přidána pouze jedna položka, a druhá (která překračuje limit) je zamítnuta.

Očekávaný výsledek: Test by měl potvrdit, že přidání více položek je zpracováno správně a že lze přidat pouze položky, které nepřekračují limit.

Test Result


TypeLimiterTest .. 6 ms
TestTypeLimit... 1 ms
TestTypeLimit... 5 ms
Group Summary
TypeLimiterTest
Tests in group: 2
Total Duration: 6 ms
Outcomes
2 Passed

WeightLimiterTest

1. Test přidávání položek až do dosažení hmotnostního limitu (*TetsWeightLimiterAddItem*)

Scénář: Tento test ověřuje, že po dosažení nastaveného hmotnostního limitu nelze do inventáře přidat žádné nové položky.

Popis: Pomocí metody CreateInventoryWithWeightLimiter je vytvořen inventář s hmotnostním limitem 6 položek.

Jsou přidány tři položky, každá o hmotnosti 2 jednotky.

Pokus o přidání čtvrté položky s hmotností 2 jednotky by měl vrátit false, protože překročí celkový limit.

Očekávaný výsledek: Test by měl potvrdit, že po dosažení hmotnostního limitu nelze přidat položku.

2. Test přidání více položek najednou (*TestWeightLimiterAddRangeItem*)

Scénář: Tento test zjišťuje, jak si WeightLimiter poradí s přidáním více položek najednou, když některé z nich lze přidat a některé z nich překročí hmotnostní limit.

Popis: Je vytvořen inventář s hmotnostním limitem 8 položek.

Jsou přidány tři položky, každá o hmotnosti 2 jednotky.

Je proveden pokus o přidání tří položek, z nichž každá má hmotnost 1 jednotky.

Je ověřeno, že do inventáře jsou přidány pouze dvě položky a třetí je odmítnuta, protože překračuje hmotnostní limit.

Očekávaný výsledek: Test by měl potvrdit, že přidání více položek je provedeno správně a že lze přidat pouze ty položky, které nepřekračují hmotnostní limit.

Test Result