Inventory Slot

Controls each individual slot of the inventory. Several slot types use this as a parent class. Mostly this class just keeps track of the item stack and UI while the Inventory Manager does most of the actual inventory work.

Inspector Variables:

```
protected Image itemImage
```

Image component to display current Item.

```
protected Image borderImage
```

Image component to display the slot border.

```
protected Image stackImage
```

Image component to display the sprite for the stack count

```
protected Text stackText
```

Text component to display the current stack amount

Public Variables:

```
public bool includeInInventory
```

Default: true
If set to true before Start() functions are called, the Inventory Manager will add this to its list of inventory slots. Set to false if being used for something with similar functionality but not actually part of the inventory, like recipe slots.

```
public ItemSO.Type type
```

Default: All
What type of items the slot can hold. Useful for equipment slots.

```
public bool interactable
```

Default: true
If false, can not be interacted with the mouse in any way.

Read Only Variables:

```
public ItemSO currentItem { get; protected set; }
```

Current item.

```
public int currentItemAmount { get; protected set; }
```

How many in the current item stack

Public Methods:

```
public int AddItemToSlot(ItemSO item, int amount)
```

If [item] matches current item, will add [amount] to current stack, otherwise it replaces current item stack with [amount] of [item].

Returns the overflow amount after adding as much to stack as possible.

```
public void SetItemInSlot(ItemSO item, int amount)
```

Hard sets item stack in slot. Ignores item types and current stack. [amount] can be set to below zero to remove items.
Use only in special cases. Currently used for crafting, removing, and swapping.

```
public bool CheckItemCompatible(ItemSO item)
```

Returns true if [item] type can is compatible with the slot.

Virtual Methods:

```
protected virtual void MouseOverChecks()
```

Called in Update() while the mouse is over the UI element.

All mouse button functionality goes here.

```
protected virtual void SetUI()
```

Called in Update().

All UI functionality goes in here.