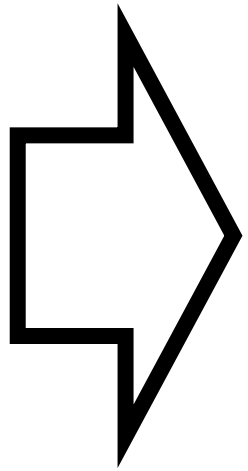
0. 동적 배열

0. 동적 배열

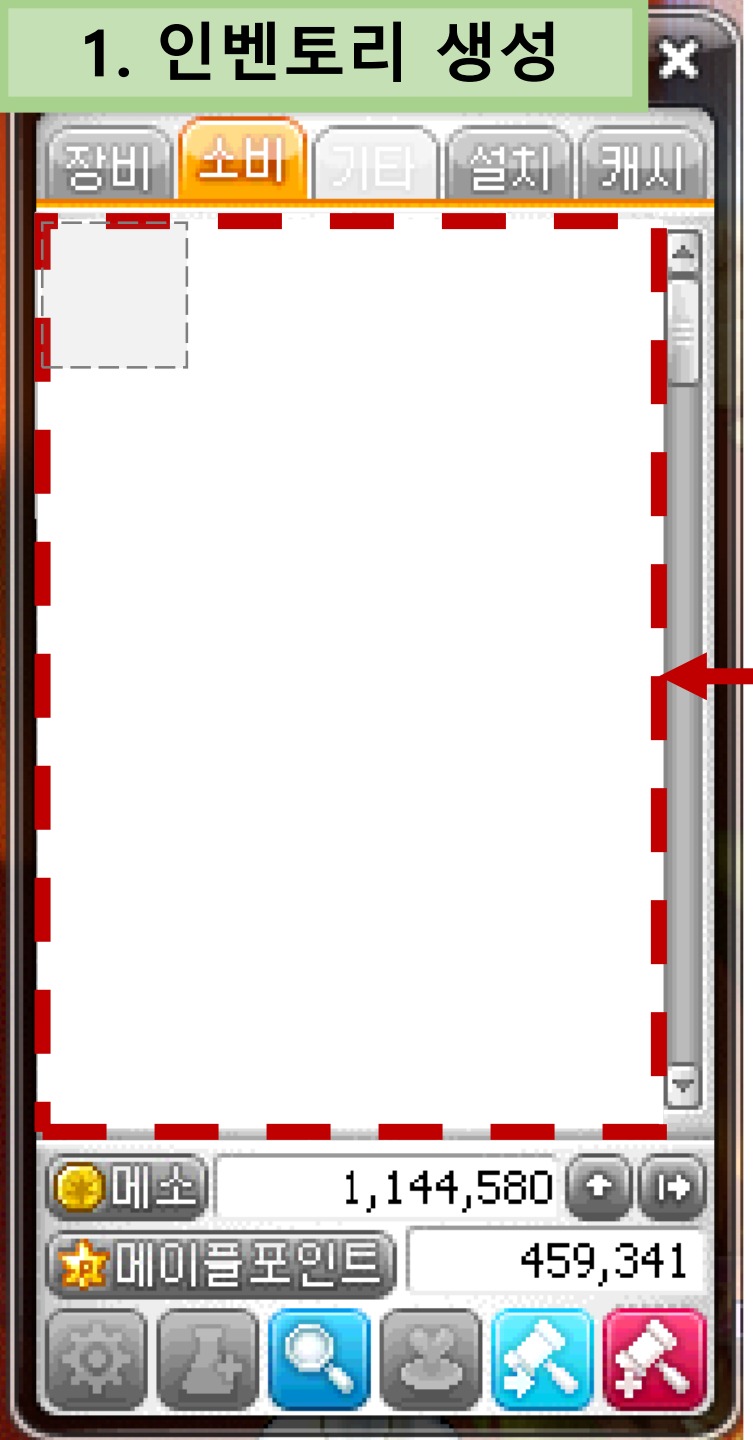

장비
소비
기타
설치
캐시
메소
1,144,580
메이플포인트
459,341




ITEM INVENTORY
장비
소비
기타
설치
캐시
34
1
1599
10
1
986
97
25
2
10
41
66
14
1
77
메소
1,144,580
메이플포인트
459,341

1. 인벤토리 생성

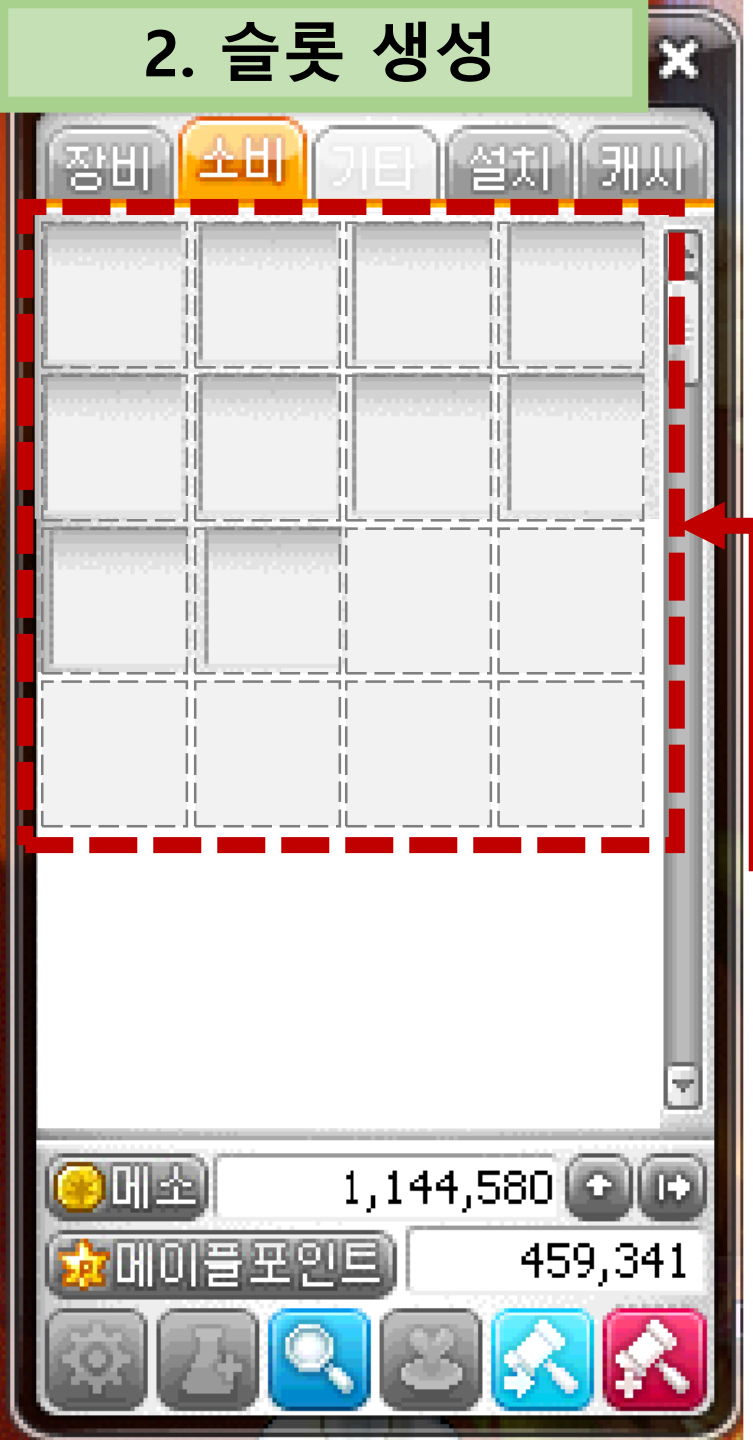
1. 인벤토리 생성




```
MyDynamicArray inventory = new MyDynamicArray();
```

2. 슬롯 생성

2. 슬롯 생성



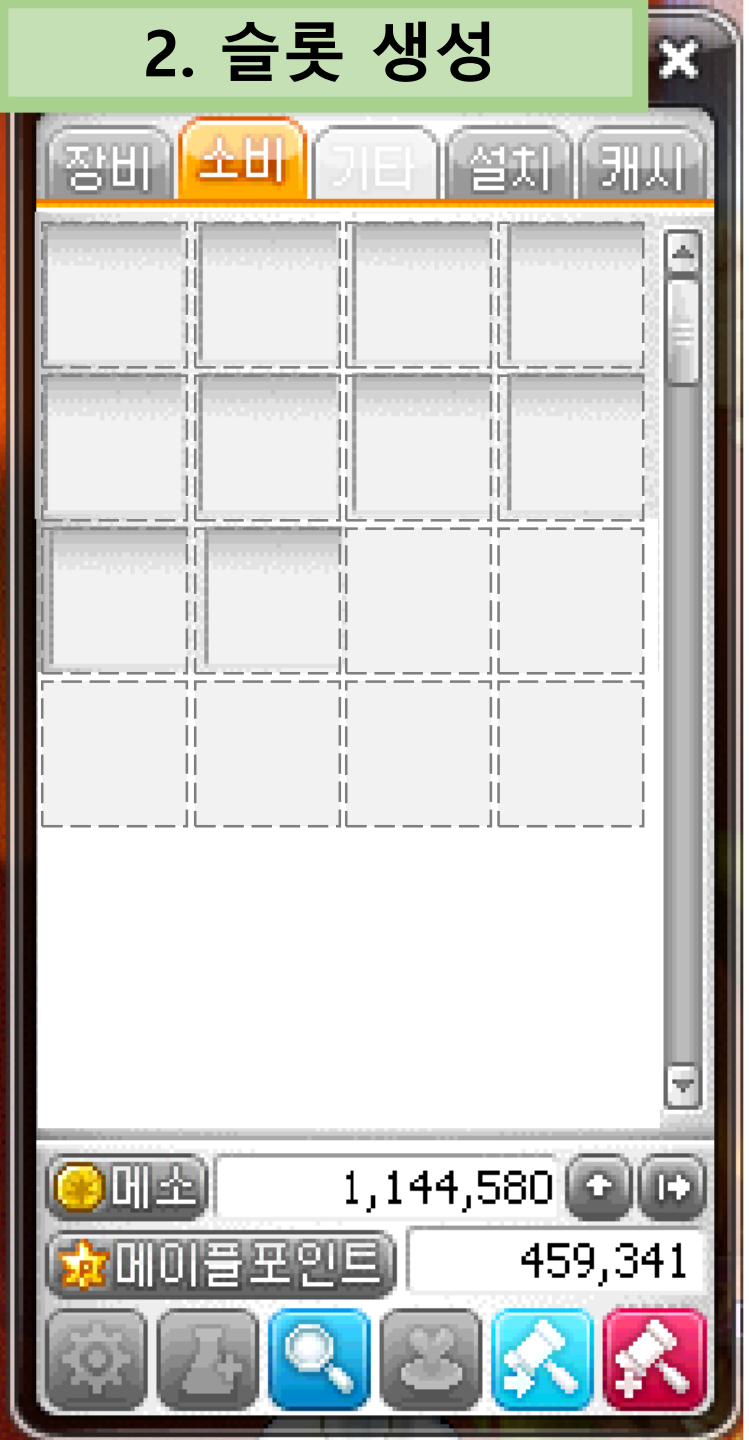

```
MyDynamicArray inventory = new MyDynamicArray();
```

```
for (int i = 0; i < 10; i++)
{
    inventory.Add(new SlotData(0, 0));
}
```

: 저장된 아이템 (빈 슬롯 포함)

: 슬롯 공간

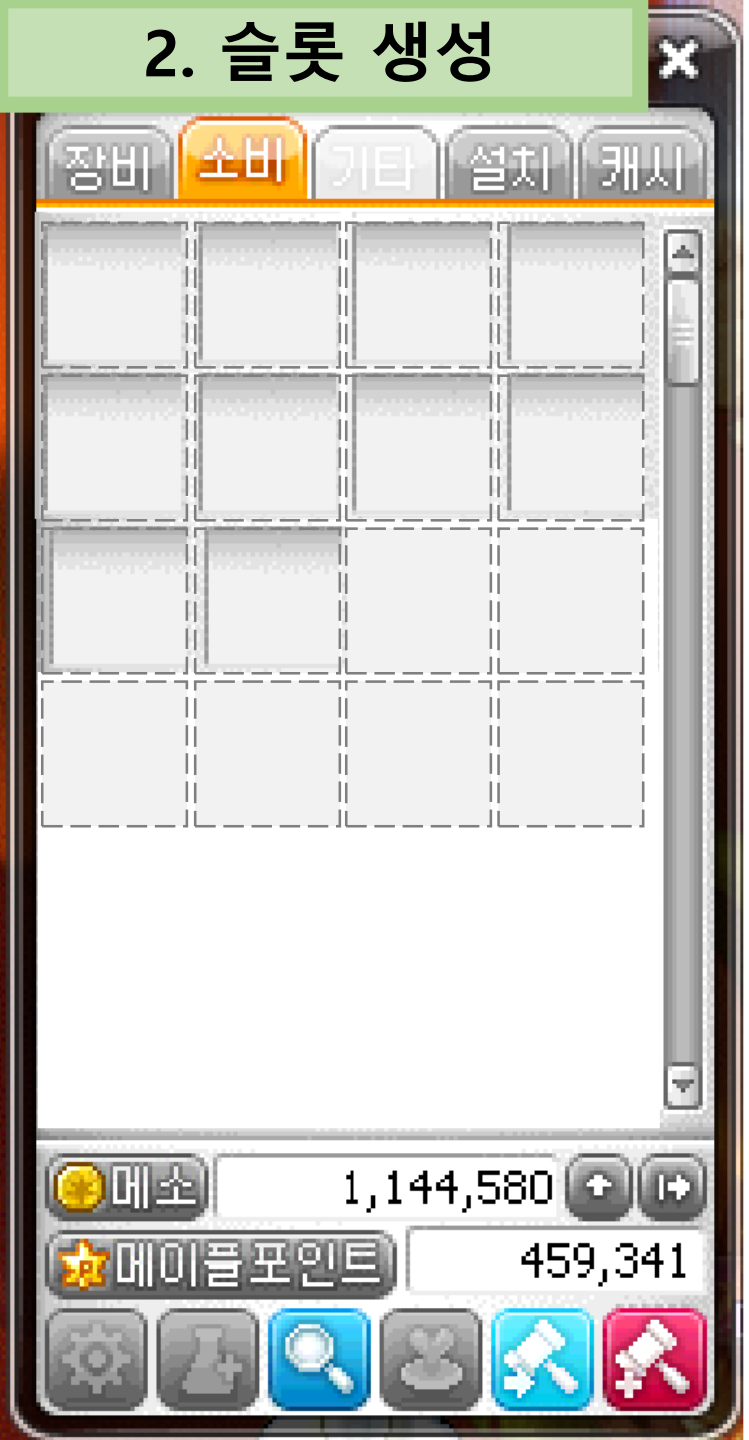
2. 슬롯 생성




```
for (int i = 0; i < 10; i++)
{
    inventory.Add(new SlotData(0, 0));
}
```

```
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
inventory.Add(new SlotData(0, 0));
```

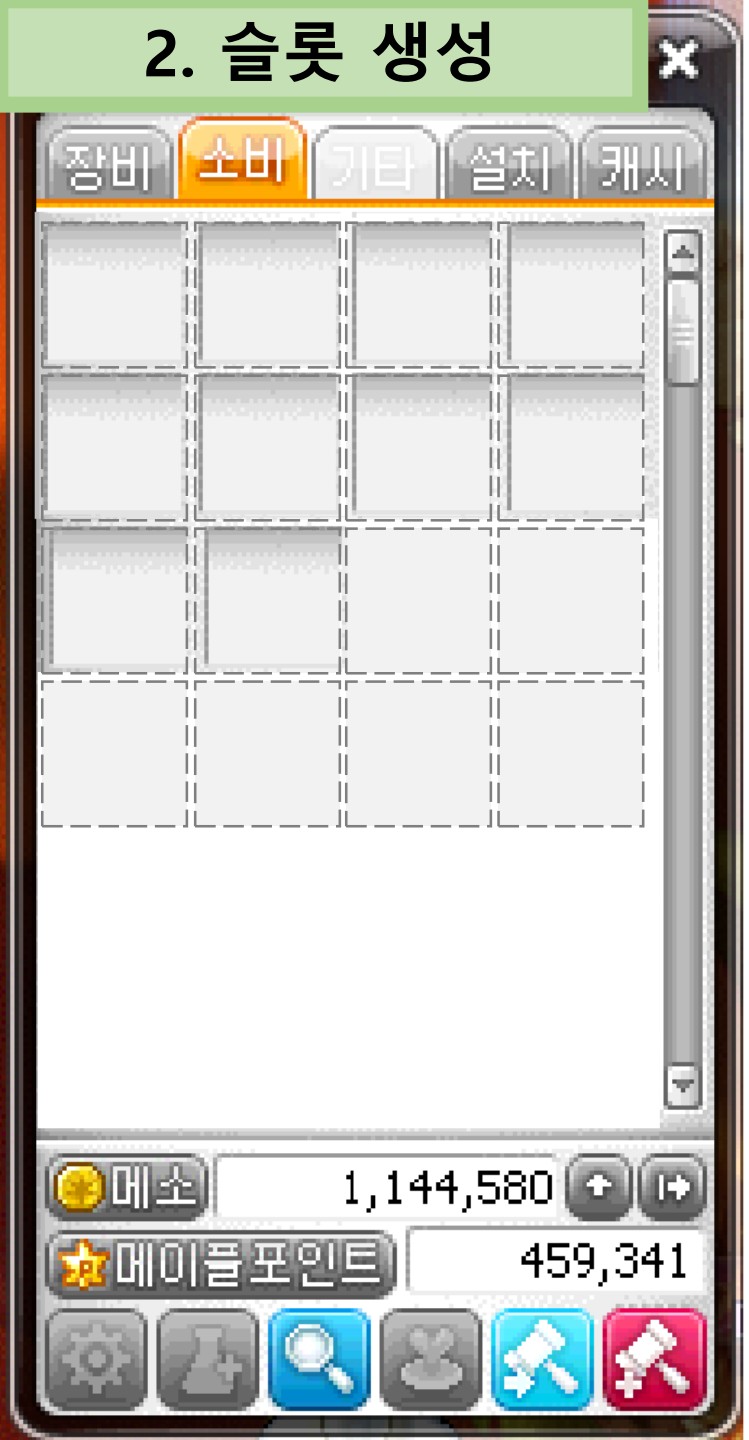
2. 슬롯 생성




```
for (int i = 0; i < 10; i++)
{
    inventory.Add(new SlotData(0, 0));
}
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

2. 슬롯 생성

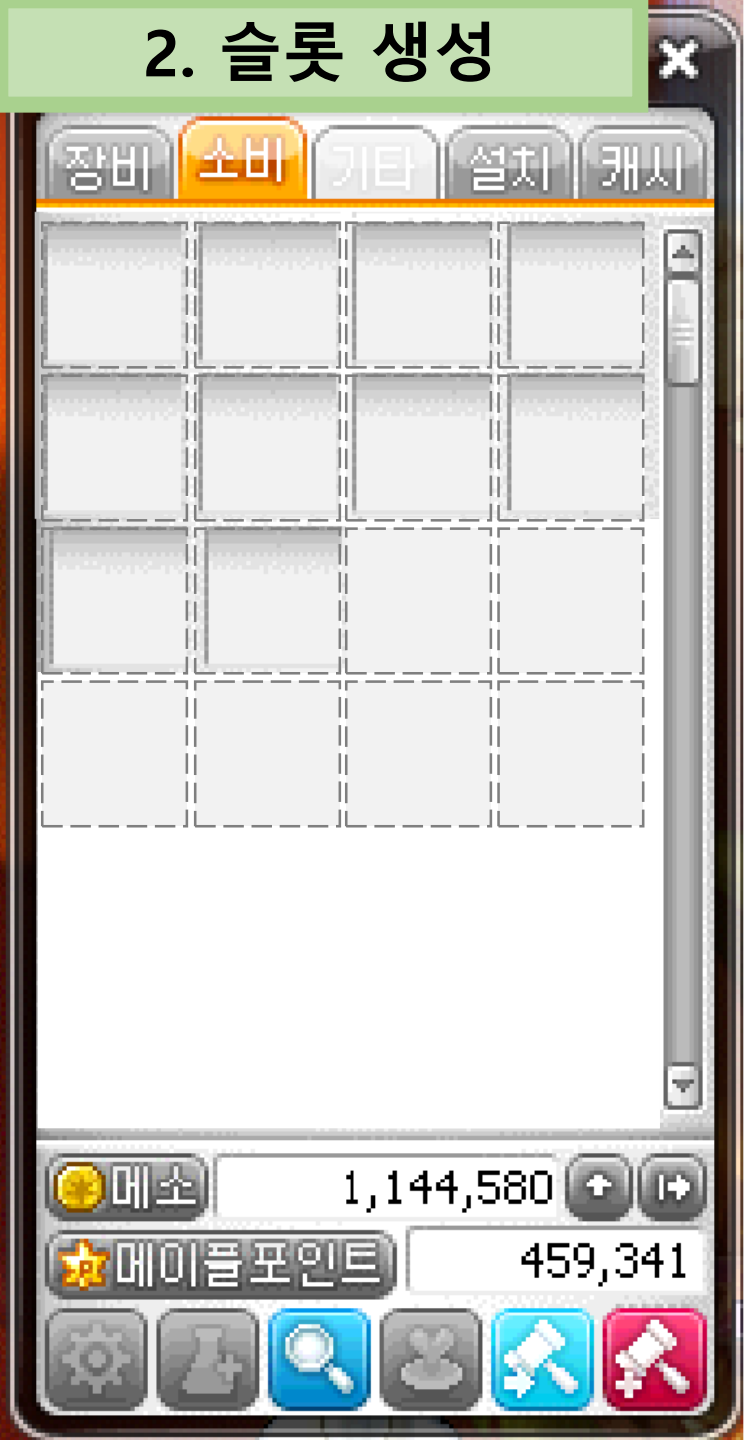



```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

2. 슬롯 생성

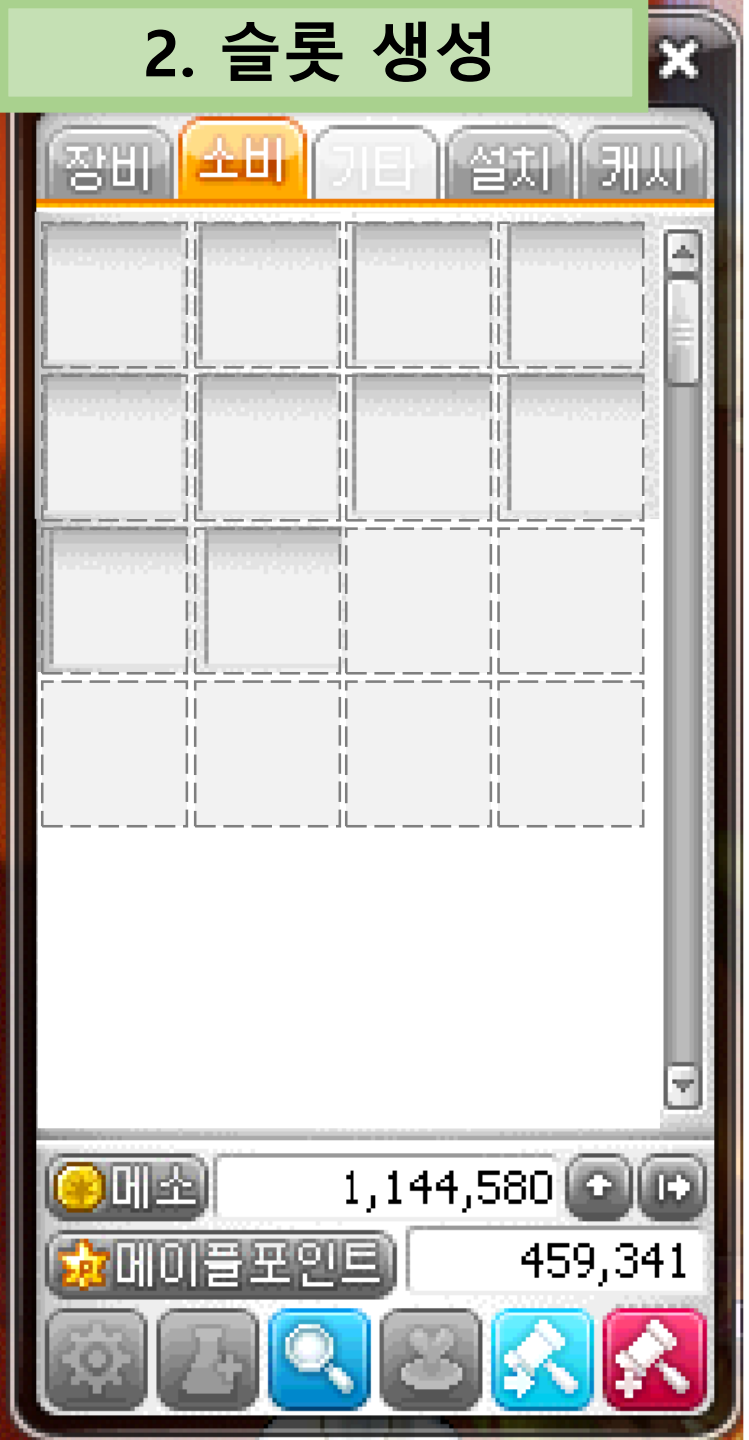



```
inventory.Add(new SlotData(0, 0));
```

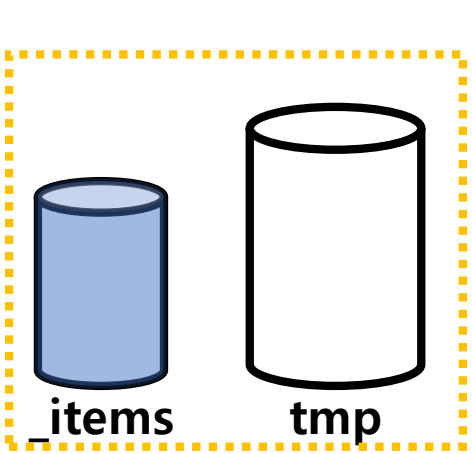
'현재 슬롯 공간'이 가득 찼을 경우 (초과했을 경우)
➔ '슬롯 공간의 개수'를 '현재 보유중인 아이템 개수'의 2배로 증가시킨 '새로운 슬롯 공간 '생성
➔ '새로운 슬롯 공간'에 현재 보유중인 아이템들을 복사
➔ '현재 슬롯 공간'을 '새로운 슬롯 공간'으로 교체

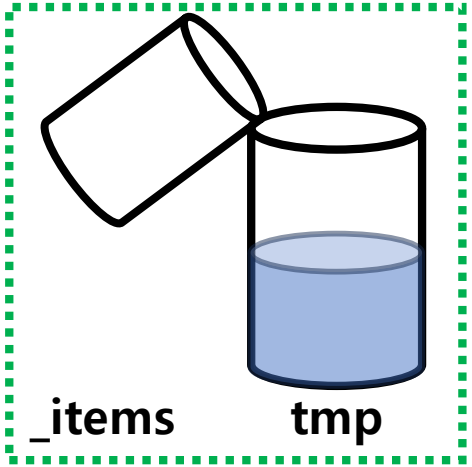
```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

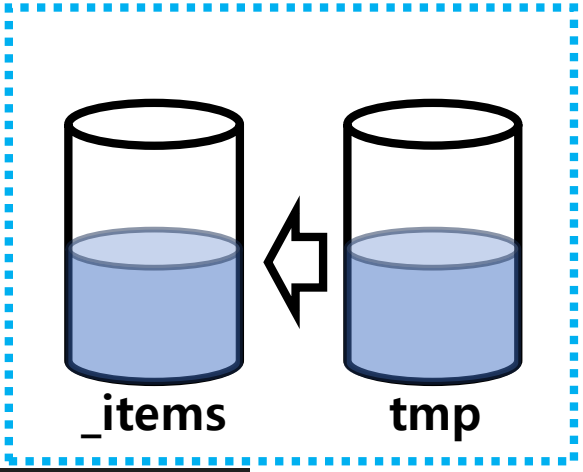
2. 슬롯 생성




```
inventory.Add(new SlotData(0, 0));
```





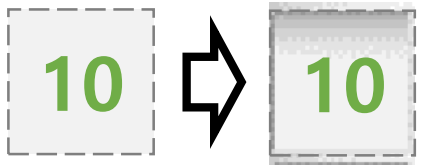



```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

2. 슬롯 생성


```
inventory.Add(new SlotData(0, 0));
```

비어 있는 슬롯에 item 정보를 저장
➔ 아이템이 늘어났으므로 '현재 보유중인 아이템 개수'에 1 증가
Ex) _count = 10인 경우, _items[10] 위치에 item 정보를 저장
_count 개수 1 증가 -> _count = 11
-> _items[10] = SlotData(0, 0) // 빈 공간
-> 10 -> 11 // _count 증가

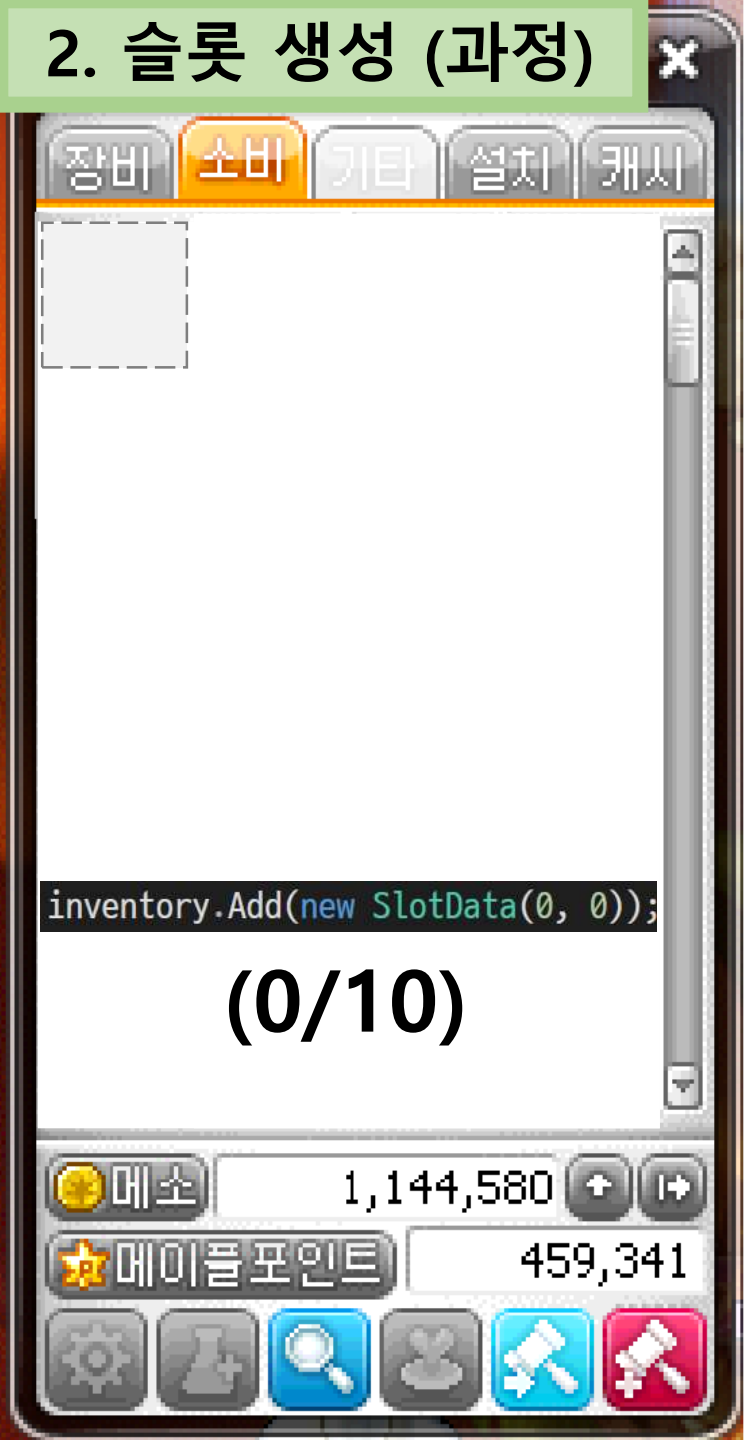



```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

2. 슬롯 생성 (과정)


장비
소비
기타
설치
캐시
inventory.Add(new SlotData(0, 0));
(0/10)
메소
1,144,580
메이플포인트
459,341

2. 슬롯 생성 (과정)




```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 0　　_items.Length = 1

2. 슬롯 생성 (과정)

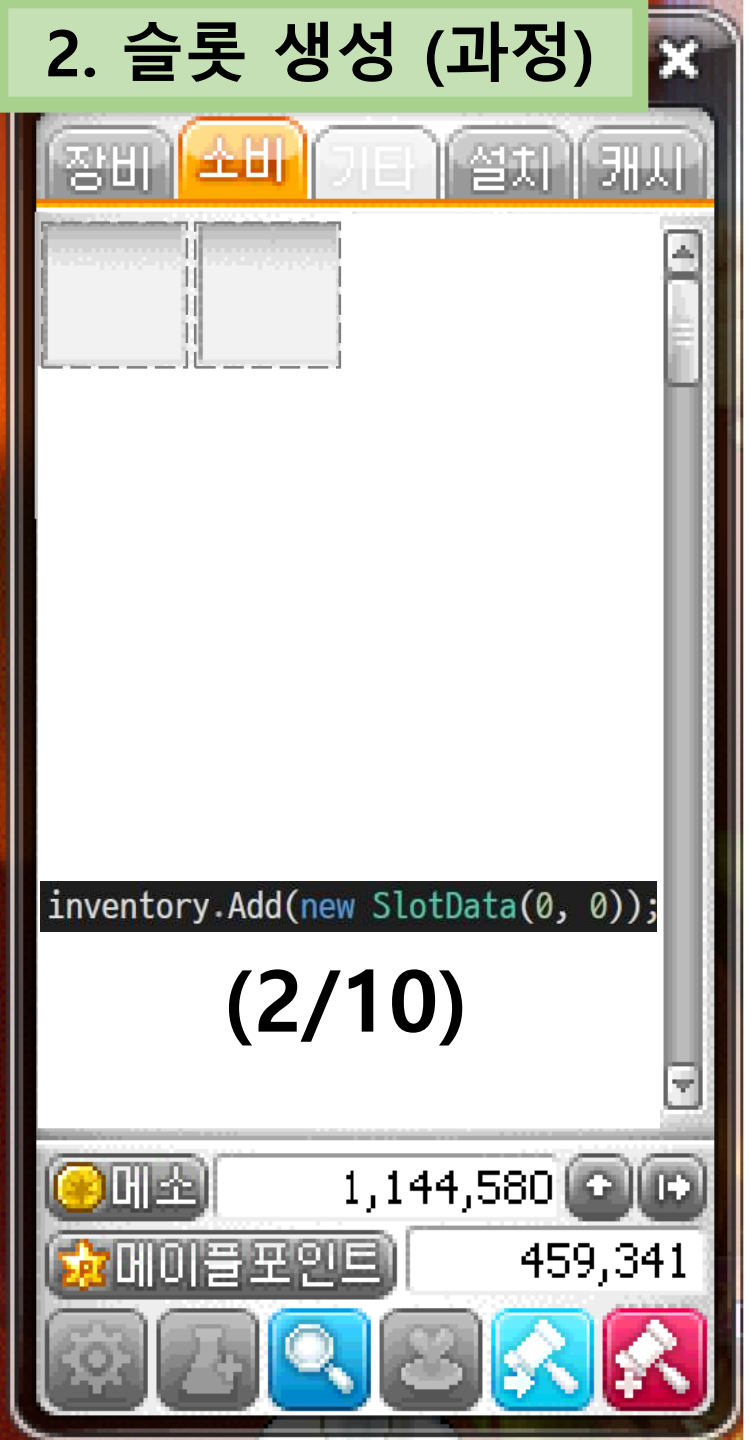

```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 1　　_items.Length = 1

2. 슬롯 생성 (과정)



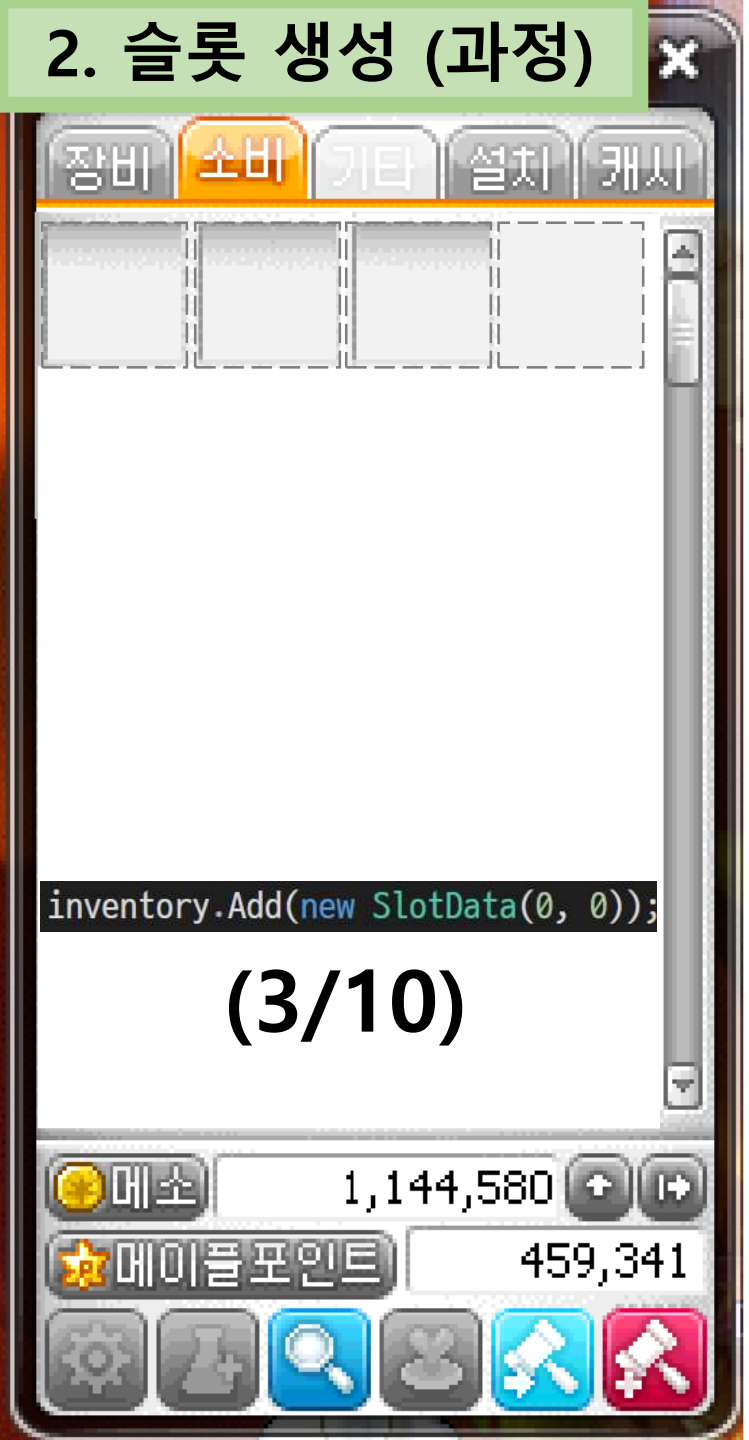

```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 2　　_items.Length = 2

2. 슬롯 생성 (과정)



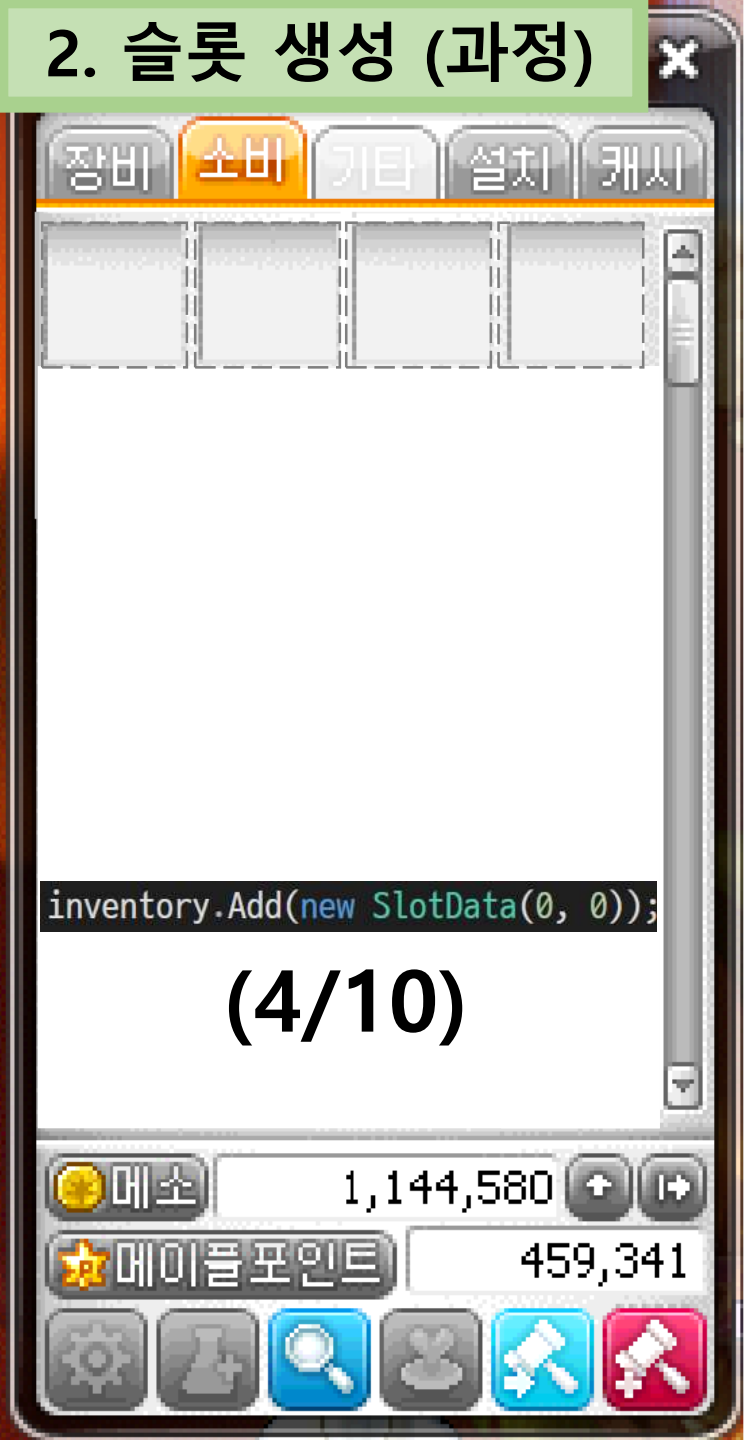

```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 3　　_items.Length = 4

2. 슬롯 생성 (과정)




```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 4　　_items.Length = 4

2. 슬롯 생성 (과정)


```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 5　　_items.Length = 8

2. 슬롯 생성 (과정)


```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 6　　_items.Length = 8

2. 슬롯 생성 (과정)


```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 7　　_items.Length = 8

2. 슬롯 생성 (과정)


```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 8　　_items.Length = 8

2. 슬롯 생성 (과정)

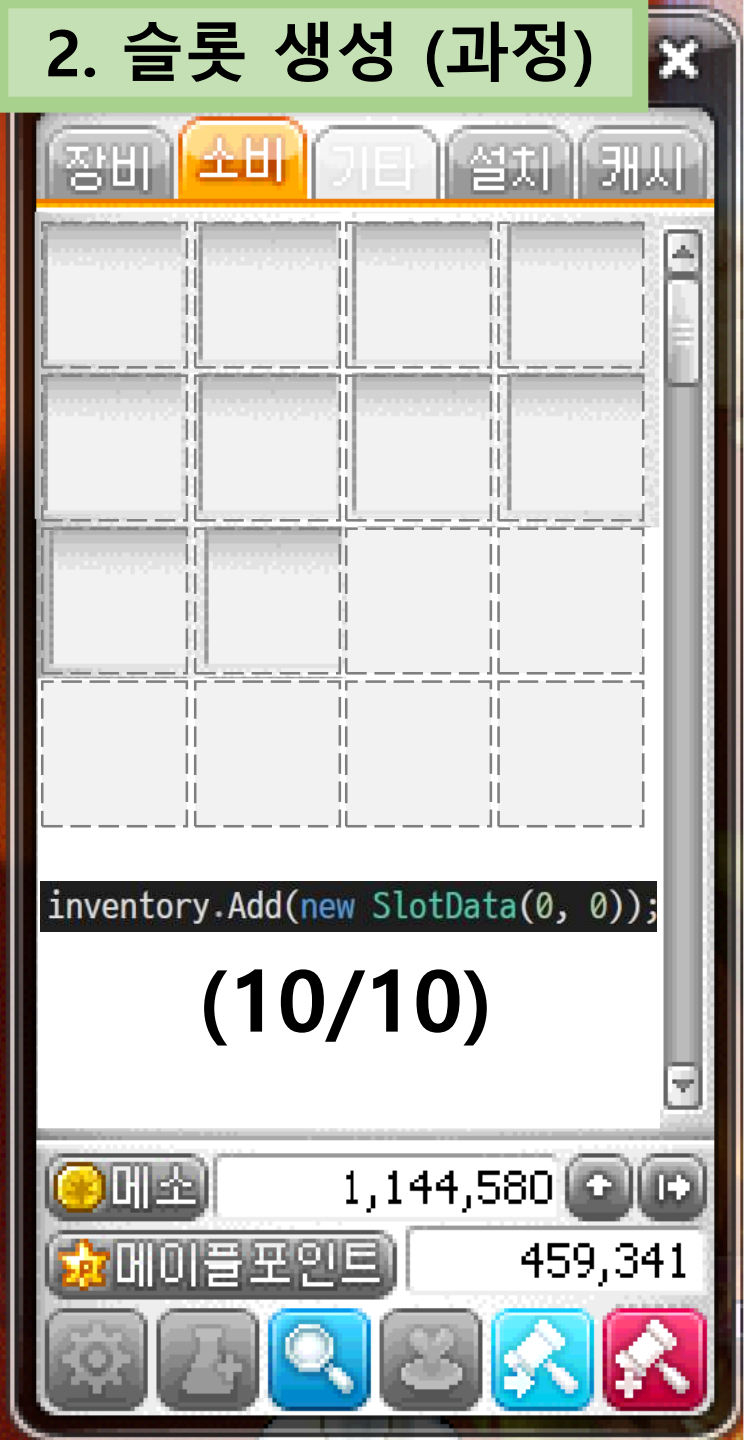

```
inventory.Add(new SlotData(0, 0));
```

```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 9　　_items.Length = 16

2. 슬롯 생성 (과정)




```
inventory.Add(new SlotData(0, 0));
```

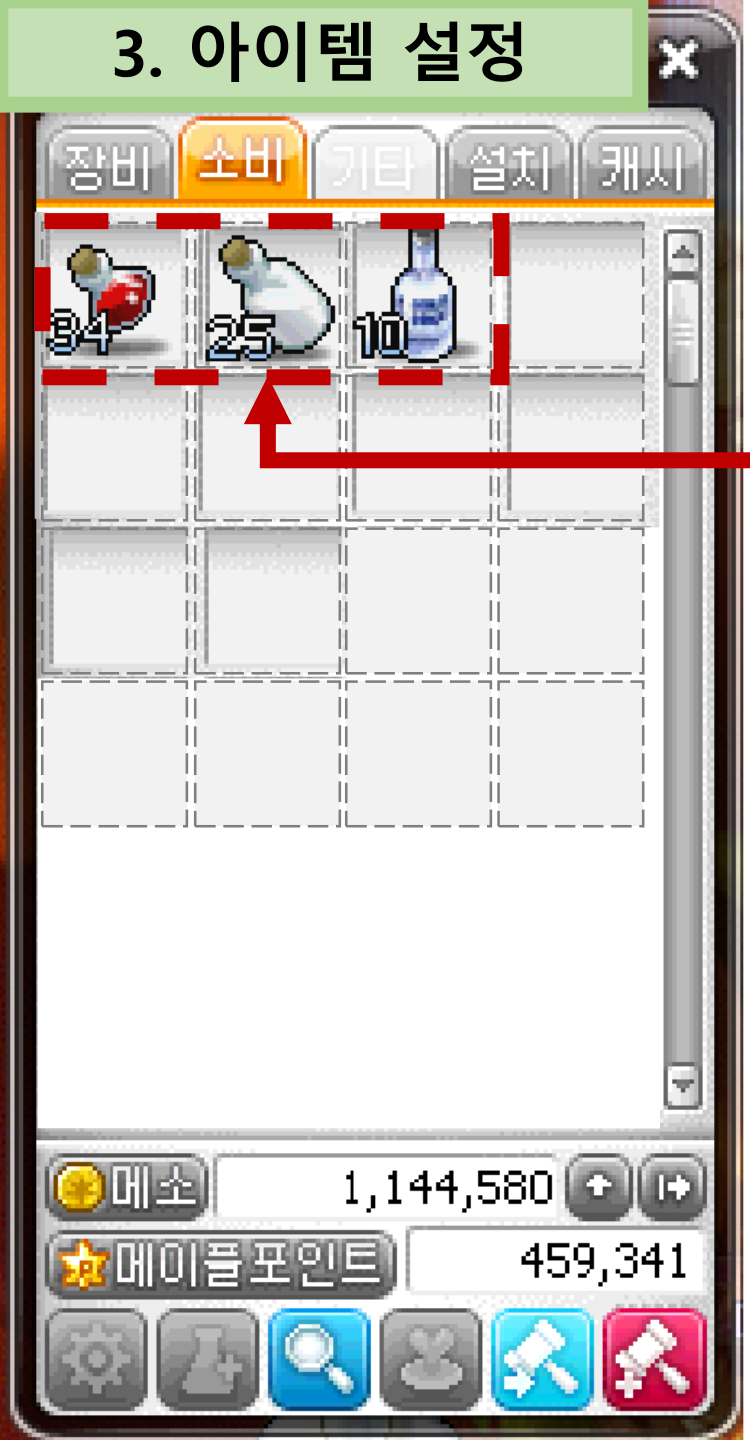
```
private int _count;
private const int DEFAULT_SIZE = 1;
private object[] _items = new object[DEFAULT_SIZE];
```

```
public void Add(object item)
{
    if (_count >= _items.Length)
    {
        object[] tmp = new object[_count * 2];
        Array.Copy(_items, tmp, _count);
        _items = tmp;
    }

    _items[_count++] = item;
}
```

_count = 10　　_items.Length = 16

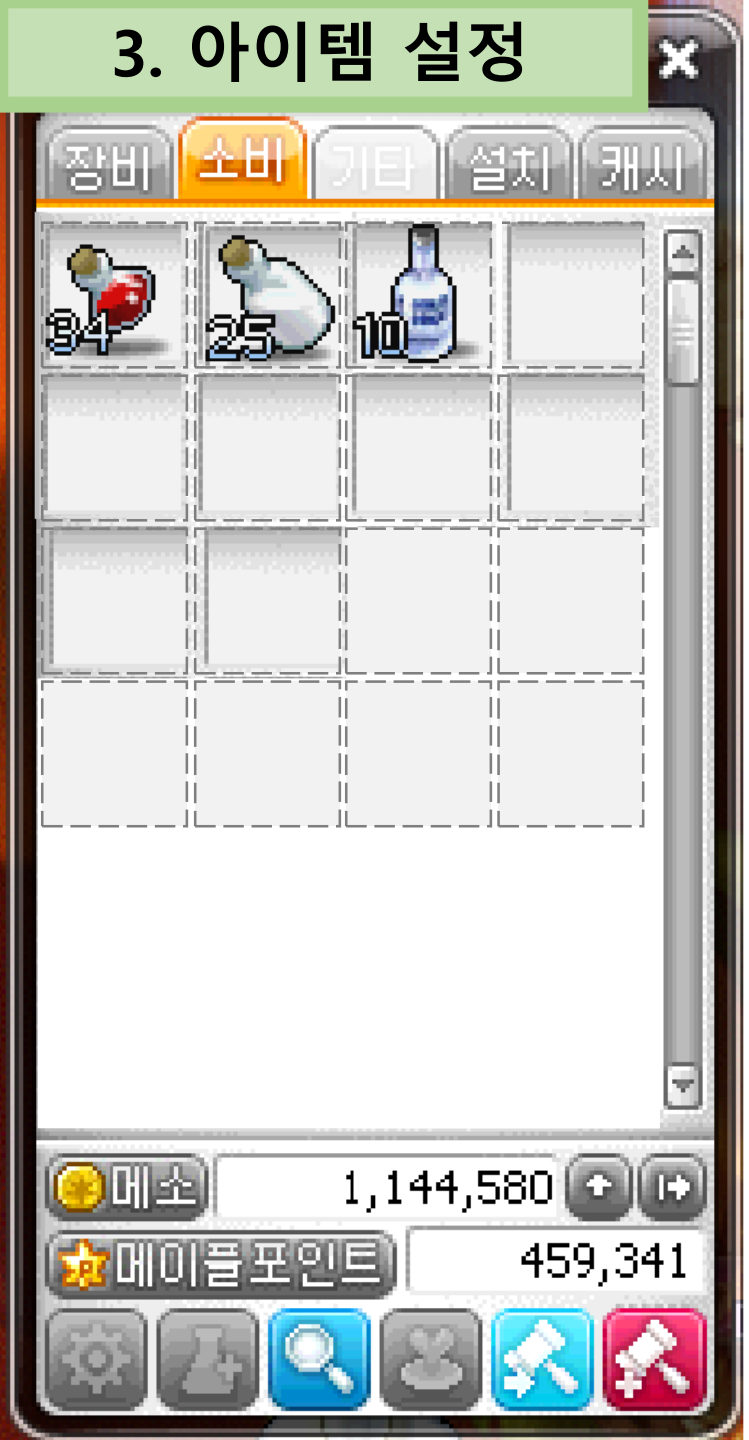
3. 아이템 설정

3. 아이템 설정




```
inventory[0] = new SlotData((int)ItemID.RedPotion, 34);
inventory[1] = new SlotData((int)ItemID.WhitePotion, 25);
inventory[2] = new SlotData((int)ItemID.Water, 10);
```

3. 아이템 설정




```
inventory[0] = new SlotData((int)ItemID.RedPotion, 34);
inventory[1] = new SlotData((int)ItemID.WhitePotion, 25);
inventory[2] = new SlotData((int)ItemID.Water, 10);
```

```
internal class MyDynamicArray
{
    public object this[int index]
    {
        get
        {
            if (index < 0 || index >= _count)
                throw new IndexOutOfRangeException();

            return _items[index];
        }
        set
        {
            if (index < 0 || index >= _count)
                throw new IndexOutOfRangeException();

            _items[index] = value;
        }
    }
```

```
enum ItemID
{
    Empty = 0,
    RedPotion = 20,
    BluePotion = 21,
    WhitePotion = 22,
    Water = 23,
    PetFood = 24,
    //...
}
```

```
class SlotData
{
    public bool isEmpty => id == 0 && num == 0;

    public int id;
    public int num;

    public SlotData(int id, int num)
    {
        this.id = id;
        this.num = num;
    }

    public int CompareTo(SlotData? other)
    {
        return this.id == other?.id && this.num == other.num ? 0 : -1;
    }
}
```

3. 아이템 설정


```
inventory[0] = new SlotData((int)ItemID.RedPotion, 34);
inventory[1] = new SlotData((int)ItemID.WhitePotion, 25);
inventory[2] = new SlotData((int)ItemID.Water, 10);
```

index : 아이템의 위치 (_items[index])

```
internal class MyDynamicArray
{
    public object this[int index]
    {
        get
        {
            if (index < 0 || index >= _count)
                throw new IndexOutOfRangeException();

            return _items[index];
        }
        set
        {
            if (index < 0 || index >= _count)
                throw new IndexOutOfRangeException();

            _items[index] = value;
        }
    }
```

index < 0 || index >= _count
(index < 0 또는 index >= 10)
: 생성된 슬롯의 범위 밖인 경우
-> 예외 발생!

_items[index] = value;
(_items[0] = 빨간포션 34개)
: _items[index]에 value 값을 저장
: (_items[0]에 빨간포션 34개 저장)

_count = 10　　_items.Length = 16

3. 아이템 설정


```
inventory[0] = new SlotData((int)ItemID.RedPotion, 34);
inventory[1] = new SlotData((int)ItemID.WhitePotion, 25);
inventory[2] = new SlotData((int)ItemID.Water, 10);
```

SlotData((int)ItemID.Water, 10)
(SlotData(23, 10))

```
enum ItemID
{
    Empty = 0,
    RedPotion = 20,
    BluePotion = 21,
    WhitePotion = 22,
    Water = 23,
    PetFood = 24,
    //...
}
```

Ex)
inventory[2].id : 23
(2번 슬롯의 아이템 ID)

inventory[2].num : 10
(2번 슬롯의 아이템 개수)

```
class SlotData
{
    public bool isEmpty => id == 0 && num == 0;

    public int id;
    public int num;

    public SlotData(int id, int num)
    {
        this.id = id;
        this.num = num;
    }

    public int CompareTo(SlotData? other)
    {
        return this.id == other?.id && this.num == other.num ? 0 : -1;
    }
}
```

3. 아이템 설정


```
inventory[0] = new SlotData((int)ItemID.RedPotion, 34);
inventory[1] = new SlotData((int)ItemID.WhitePotion, 25);
inventory[2] = new SlotData((int)ItemID.Water, 10);
```

(SlotData)Inventory[0].id = RedPotion
(SlotData)Inventory[0].num = 34

(SlotData)Inventory[1].id = WhitePotion
(SlotData)Inventory[1].num = 25

(SlotData)Inventory[2].id = Water
(SlotData)Inventory[2].num = 10

```
enum ItemID
{
    Empty = 0,
    RedPotion = 20,
    BluePotion = 21,
    WhitePotion = 22,
    Water = 23,
    PetFood = 24,
    //...
}
```

3. 아이템 설정


```
Console.WriteLine("인벤토리 정보 :");
for (int i = 0; i < inventory.Count; i++)
{
    Console.WriteLine($"슬롯 {i} : [{(ItemID)((SlotData)inventory[i]).id}] , [{((SlotData)inventory[i]).num}]");
}
```

```
인벤토리 정보 :
슬롯 0 : [RedPotion] , [34]
슬롯 1 : [WhitePotion] , [25]
슬롯 2 : [Water] , [10]
슬롯 3 : [Empty] , [0]
슬롯 4 : [Empty] , [0]
슬롯 5 : [Empty] , [0]
슬롯 6 : [Empty] , [0]
슬롯 7 : [Empty] , [0]
슬롯 8 : [Empty] , [0]
슬롯 9 : [Empty] , [0]
```

```
enum ItemID
{
    Empty = 0,
    RedPotion = 20,
    BluePotion = 21,
    WhitePotion = 22,
    Water = 23,
    PetFood = 24,
    //...
}
```

(SlotData)Inventory[0].id = RedPotion
(SlotData)Inventory[0].num = 34

(SlotData)Inventory[1].id = WhitePotion
(SlotData)Inventory[1].num = 25

(SlotData)Inventory[2].id = Water
(SlotData)Inventory[2].num = 10

(SlotData)Inventory[2].id = Empty
(SlotData)Inventory[2].num = 0

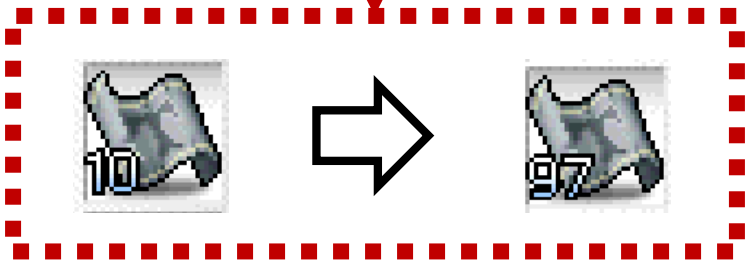
inventory.Count = 10

_count = 10　　_items.Length = 16

4. 동적 배열 기능 사용

4. 동적 배열 기능 사용


```
// todo -> 지도 87개를 획득
// 1. 지도 87개가 들어갈 수 있는 슬롯을 찾아야함.
int availableSlotIndex = inventory.FindIndex(slotData => ((SlotData)slotData).isEmpty ||
                                                         (((SlotData)slotData).id == (int)ItemID.Map &&
                                                          ((SlotData)slotData).num <= 99 - 87));

// 2. 해당 슬롯의 아이템 갯수에다가 내가 추가하려는 갯수를 더한 만큼의 수정예상값을 구함.
int expected = ((SlotData)inventory[availableSlotIndex]).num + 87;

// 3. 새로운 아이템 데이터를 만들어서 슬롯 데이터를 교체 해줌.
SlotData newSlotData = new SlotData((int)ItemID.Map, expected);
inventory[availableSlotIndex] = newSlotData;
```



4. 동적 배열 기능 사용


```
// todo -> 지도 87개를 획득
// 1. 지도 87개가 들어갈 수 있는 슬롯을 찾아야함.
int availableSlotIndex = inventory.FindIndex(slotData => ((SlotData)slotData).isEmpty ||
                                                         (((SlotData)slotData).id == (int)ItemID.Map &&
                                                          ((SlotData)slotData).num <= 99 - 87));

// 2. 해당 슬롯의 아이템 갯수에다가 내가 추가하려는 갯수를 더한 만큼의 수정예상값을 구함.
int expected = ((SlotData)inventory[availableSlotIndex]).num + 87;

// 3. 새로운 아이템 데이터를 만들어서 슬롯 데이터를 교체 해줌.
SlotData newSlotData = new SlotData((int)ItemID.Map, expected);
inventory[availableSlotIndex] = newSlotData;
```

availableSlotIndex :
인벤토리 내에 '특정 조건'을 만족하는 슬롯의 index 위치 값
(지도 87개가 들어갈 수 있는 슬롯의 index 위치 값)

'특정 조건' :
((SlotData)slotData).isEmpty || // 빈 슬롯이거나
(((SlotData)slotData).id == (int)Item.Map && // id가 Map이면서
((SlotData)slotData).num <= 99 – 87) // 개수가 12개 이하일 경우

4. 동적 배열 기능 사용


```
// todo -> 지도 87개를 획득
// 1. 지도 87개가 들어갈 수 있는 슬롯을 찾아야함.
int availableSlotIndex = inventory.FindIndex(slotData => ((SlotData)slotData).isEmpty ||
                                                         (((SlotData)slotData).id == (int)ItemID.Map &&
                                                          ((SlotData)slotData).num <= 99 - 87));

// 2. 해당 슬롯의 아이템 갯수에다가 내가 추가하려는 갯수를 더한 만큼의 수정예상값을 구함.
int expected = ((SlotData)inventory[availableSlotIndex]).num + 87;

// 3. 새로운 아이템 데이터를 만들어서 슬롯 데이터를 교체 해줌.
SlotData newSlotData = new SlotData((int)ItemID.Map, expected);
inventory[availableSlotIndex] = newSlotData;
```

'특정 조건' :
((SlotData)slotData).isEmpty || // 빈 슬롯이거나
(((SlotData)slotData).id == (int)Item.Map && // id가 Map이면서
((SlotData)slotData).num <= 99 – 87) // 개수가 12개 이하일 경우

inventory.FindIndex(slotData => '특정 조건') = 5

```
public int FindIndex(Predicate<object> match)
{
    for (int i = 0; i < _count; i++)
    {
        if (match(_items[i]))
            return i;
    }
    return -1;
}
```

availableSlotIndex = 5

4. 동적 배열 기능 사용


```
// todo -> 지도 87개를 획득
// 1. 지도 87개가 들어갈 수 있는 슬롯을 찾아야함.
int availableSlotIndex = inventory.FindIndex(slotData => ((SlotData)slotData).isEmpty ||
                                                         (((SlotData)slotData).id == (int)ItemID.Map &&
                                                          ((SlotData)slotData).num <= 99 - 87));

// 2. 해당 슬롯의 아이템 갯수에다가 내가 추가하려는 갯수를 더한 만큼의 수정예상값을 구함.
int expected = ((SlotData)inventory[availableSlotIndex]).num + 87;

// 3. 새로운 아이템 데이터를 만들어서 슬롯 데이터를 교체 해줌.
SlotData newSlotData = new SlotData((int)ItemID.Map, expected);
inventory[availableSlotIndex] = newSlotData;
```

expected : 아이템 개수 수정 예상 값

expected = (SlotData)inventory[availableSlotIndex].num + 87

expected = (SlotData)Inventory[5].num + 87

➔ (SlotData)Inventory[5].num = 10

➔ 10 + 87 = 97

(SlotData)Inventory[5].num = 10

availableSlotIndex = 5

expected = 97

4. 동적 배열 기능 사용


```
// todo -> 지도 87개를 획득
// 1. 지도 87개가 들어갈 수 있는 슬롯을 찾아야함.
int availableSlotIndex = inventory.FindIndex(slotData => ((SlotData)slotData).isEmpty ||
                                                         (((SlotData)slotData).id == (int)ItemID.Map &&
                                                          ((SlotData)slotData).num <= 99 - 87));

// 2. 해당 슬롯의 아이템 갯수에다가 내가 추가하려는 갯수를 더한 만큼의 수정예상값을 구함.
int expected = ((SlotData)inventory[availableSlotIndex]).num + 87;

// 3. 새로운 아이템 데이터를 만들어서 슬롯 데이터를 교체 해줌.
SlotData newSlotData = new SlotData((int)ItemID.Map, expected);
inventory[availableSlotIndex] = newSlotData;
```

newSlotData : 새로운 아이템 생성
newSlotData = new SlotData((int)ItemID.Map, expected)
newSlotData = new SlotData(25, 97) // 지도 97개

inventory[availableSlotIndex] = newSlotData
inventory[5] = 지도 97개

```
enum ItemID
{
    Empty = 0,
    RedPotion = 20,
    BluePotion = 21,
    WhitePotion = 22,
    Water = 23,
    PetFood = 24,
    Map = 25,
    //...
}
```

availableSlotIndex = 5

expected = 97

newSlotData = 지도 97개

QnA

조영민 강사님의 Main 함수를 일부분 수정한 내용입니다.
자세한 문의는 DM 부탁드립니다.🙏
-김영호-