```
public abstract class BaseBossState<T>
{
   public abstract void EnterState(T boss,Transform player);
   public abstract void UpdateState(T boss,Transform player);

}

public class GroundBoss : EnemyEntity
{


   float floor_Distance = 1.95f;
   Vector3 floor_Dir = Vector3.down;
   Vector2 floor_Size = new Vector2(1f, 0.3f);
   Vector2 lastVelocity;

   GameObject townPortal;

   BaseBossState<GroundBoss> currentState;
   public GroundBossNormalAttackState NormalAttackState = new
GroundBossNormalAttackState();
   public GroundBossBounceAttackState BounceAttackState = new
GroundBossBounceAttackState();
   public GroundBossRushAttackState RushAttackState = new
GroundBossRushAttackState();
   public GroundBossShootAttackState ShootAttackState = new
GroundBossShootAttackState();
   public GroundBossVolcanoAttackState VolcanoAttackState = new
GroundBossVolcanoAttackState();
   GroundBossDropItem groundBossDropItem;

   public override void Awake()
   {
      base.Awake();
      player =
GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerController>();
      groundBossDropItem = new GroundBossDropItem();
   }
   public override void Start()
   {
      base.Start();

      enemyStat.curHealth = enemyStat.maxHealth;
   }
   private void Update()
   {
      StateChecker();
      currentState?.UpdateState(this,player.transform);
```

```
    }

    public override IEnumerator Spawn()
    {
        Debug.Log("스폰");
        SwitchState(NormalAttackState);
        yield return null;

    }

    void StateChecker()
    {
        GimmickTimer += Time.deltaTime;
        IsGround = MyUtils.WhatFloor(transform.position, floor_Distance, floor_Dir, floor_Size,
"Ground");
        LookDir = player.transform.position.x - transform.position.x > 0 ? 1 : -1;
        if (IsGround)
            sprite.flipX = LookDir > 0 ? true : false;
    }

    public override IEnumerator Die()
    {
        currentState = null;
        Color color = Color.white;
        while (true)
        {
            color.a -= Time.deltaTime*0.5f;
            sprite.color = color;
            if (color.a < 0)
            {
                groundBossDropItem.DropItems(transform);
                townPortal = GameObject.FindGameObjectWithTag("Portal");
                townPortal.transform.GetChild(0).gameObject.SetActive(true);
                gameObject.SetActive(false);
                break;
            }
            yield return null;
        }

    }

    public void SwitchState(BaseBossState<GroundBoss> state)
    {
        currentState = state;
    }

    public void PatternSwitch()
    {
        int pattern;
        if (Phase2Check())
            pattern = Random.Range(1,6);
        else
            pattern = Random.Range(1,3);
        switch (pattern)
        {
            case 1:
                SwitchState(ShootAttackState);
                break;
            case 2:
                SwitchState(RushAttackState);
                break;
            case 3:
                SwitchState(BounceAttackState);
                break;
            case 4:
            case 5:
                SwitchState(VolcanoAttackState);
                break;
        }
        currentState.EnterState(this, player.transform);

    }


    public override void OnTriggerEnter2D(Collider2D collision)
    {
        base.OnTriggerEnter2D(collision);
        if (collision.transform.CompareTag("Wall"))
        {
            if (rigid.velocity.y > 0)
                lastVelocity = rigid.velocity;

            Vector3 reflectDir = Vector3.Reflect(lastVelocity.normalized,
collision.transform.position);
            rigid.velocity = reflectDir.normalized * Mathf.Max(0f, 1f);
        }

    }
    private void OnDrawGizmos()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireCube(transform.position + (floor_Distance * floor_Dir), floor_Size);
    }

}
```

```
public abstract class EnemyEntity : MonoBehaviour
{
    public float LookDir { get; set; }
    public bool IsGround { get; set; }
    public float GimmickTimer { get; set; }
    public EnemyStatsSO enemyStat;
    bool isOnHit;
    int hitTimerCount;
    Material glowMat;
    WaitForSeconds colorBlinkTime;
    BossHealthUI healthUI;
    protected SpriteRenderer sprite;
    protected Rigidbody2D rigid;
    protected PlayerController player;
    public virtual void Awake()
    {
        rigid = GetComponent<Rigidbody2D>();

        sprite = GetComponent<SpriteRenderer>();
        healthUI = GetComponentInChildren<BossHealthUI>();
        glowMat = GetComponent<SpriteRenderer>().material;
        colorBlinkTime = new WaitForSeconds(0.1f);
    }
    public virtual void Start()
    {
        healthUI.HealthUI(enemyStat.curHealth, enemyStat.maxHealth);
        StartCoroutine(healthUI.BossText(enemyStat.name, Spawn()));
    }
    public virtual void OnDamaged(float damage, Color color,float fontSize)
    {
        enemyStat.curHealth -= damage;
        DamageText.Create(transform.position, damage, color, fontSize);
        healthUI.HealthUI(enemyStat.curHealth, enemyStat.maxHealth);
        hitTimerCount = 0;
        BossSpriteChange();

        if (enemyStat.curHealth <= 0)
        {
            enemyStat.curHealth = 0;
            Debug.Log("보스사망");
            StartCoroutine(Die());
        }
        else
        {
            if (!isOnHit) StartCoroutine(DamagedEffect());
        }
    }
```

```
void BossSpriteChange()
{
    if (enemyStat.curHealth < enemyStat.maxHealth / 2)
    {
        Debug.Log("페이즈2 컬러변환");
        Color phase2Color = new Color(1, 0.5f, 0.5f, 1);
        sprite.color = phase2Color;
    }

}

IEnumerator DamagedEffect()
{
    isOnHit = true;
    while (hitTimerCount < 3)
    {
        Color hitColor = new Color(1, 1, 1, 0);
        glowMat.SetColor("_GlowColor", hitColor);
        yield return colorBlinkTime;
        hitColor = new Color(0, 0, 0, 0);
        glowMat.SetColor("_GlowColor", hitColor);
        yield return colorBlinkTime;
        hitTimerCount++;
    }
    isOnHit = false;
}

public void CameraShaking()
{
    StartCoroutine(CameraShake.ShakeCoroutine(0.1f));
}
public bool Phase2Check()
{
    bool phase2;

    if (enemyStat.curHealth > enemyStat.maxHealth / 2)
        phase2 = false;
    else
        phase2 = true;

    return phase2;
}
public abstract IEnumerator Spawn();

public abstract IEnumerator Die();

public virtual void OnTriggerEnter2D(Collider2D collision)
```

```
    {
        if (collision.CompareTag("Melee"))
        {
            WeaponController weapon = collision.GetComponentInParent<WeaponController>();


OnDamaged(weapon.HitDamage(),weapon.currentWeapon.currentDamageTxtColor,weapon
.currentWeapon.currentFontSize);

        }
        else if (collision.CompareTag("Magic"))
        {
            Projectile projectile = collision.GetComponent<Projectile>();
            OnDamaged(projectile.HitDamage(),
projectile.weapon.currentWeapon.currentDamageTxtColor,
projectile.weapon.currentWeapon.currentFontSize);
            collision.gameObject.SetActive(false);

        }
    }

}

public class UIItem : MonoBehaviour, IPointerClickHandler, IBeginDragHandler,
IEndDragHandler, IDragHandler, IDropHandler, IPointerExitHandler, IPointerMoveHandler
{
    [SerializeField] Image itemImage;
    [SerializeField] TMP_Text quantityTxt;
    [SerializeField] TMP_Text UsingTxt;
    [SerializeField] Image borderImage;
    public event Action<UIItem> PointerRightClick,PointerShiftRightClick, BeginDrag,
EndDrag, Drop, PointerExit;
    public event Func<UIItem, UIDescription> PointerMove;

    public bool empty = true;

    public ItemSO item;
    public int Quantity { get; set; }
    private void Awake()
    {
        ResetData();
    }

    public void ResetData()
    {
        itemImage.enabled = false;
        quantityTxt.gameObject.SetActive(false);
        item = null;
    empty = true;
}

public void SetData(Sprite itemImage, int quantity, ItemSO item)
{

    this.itemImage.sprite = itemImage;
    quantityTxt.text = quantity + "";
    this.Quantity = quantity;
    empty = false;
    this.item = item;
    this.itemImage.enabled = true;
    quantityTxt.gameObject.SetActive(true);
}

public void SetData(Sprite itemImage, int quantity)
{
    SetData(itemImage, quantity, null);
}
public void SetData(ItemSO item,int quantity)
{
    SetData(item.itemImage, quantity, item);
}

public void SellItem()
{
    GameManager.instance.Money += item.SellCost;
    Updatequantity(-1);
}
public void Updatequantity(int quantity)
{
    quantity = int.Parse(quantityTxt.text) + quantity;
    this.Quantity = quantity;
    if (quantity <= 0)
    {
        ResetData();
    }
    quantityTxt.text = quantity + "";

}

public void InitUsingTxt(int num)
{
    itemImage.enabled = false;
    UsingTxt.gameObject.SetActive(true);
    UsingTxt.text = num + "";
}
```

```
    public void ToggleQuantityTxt(bool val)
    {
        quantityTxt.gameObject.SetActive(val);
    }

    public void ItemSelected(bool val)
    {
        if (val)
        {
            borderImage.color = Color.yellow;
        }
        else
        {
            borderImage.color = new Color(1, 1, 1, 0.2f);
        }

    }

    public void OnPointerClick(PointerEventData eventData)
    {
        if (empty) return;
        SoundManager.instance.PlaySound(SoundType.InventoryItemClick);
        if (eventData.button == PointerEventData.InputButton.Right)
        {
            if (Input.GetKey(KeyCode.LeftShift))
                PointerShiftRightClick?.Invoke(this);
            else
                PointerRightClick?.Invoke(this);

        }
        //if(eventData.button == PointerEventData.InputButton.Right &&
Input.GetKeyDown(KeyCode.LeftShift))
        //    Debug.Log("쉬프트 우클릭");
    }

    public void OnBeginDrag(PointerEventData eventData)
    {
        if (empty) return;
        BeginDrag?.Invoke(this);
    }

    public void OnDrag(PointerEventData eventData)
    {

    }

    public void OnDrop(PointerEventData eventData)
    {
        Drop?.Invoke(this);
    }

    public void OnEndDrag(PointerEventData eventData)
    {
        EndDrag?.Invoke(this);
    }
    public void OnPointerExit(PointerEventData eventData)
    {
        PointerExit?.Invoke(this);
    }

    public void OnPointerMove(PointerEventData eventData)
    {
        if (empty) return;
        UIDescription description = PointerMove?.Invoke(this);
        if (description != null)
            description.transform.position = eventData.position;
    }
}

public class UIInventory : MonoBehaviour
{
    [HideInInspector]
    public List<UIItem> equips = new List<UIItem>();
    [HideInInspector]
    public List<UIItem> items = new List<UIItem>();
    List<UIItem> usings = new List<UIItem>();
    List<UIItem> tmpList = new List<UIItem>();

    public event Action<UIItem> UseItemAction;
    public event Action CurrentEquipEffectsCheck;
    [SerializeField] RectTransform equipBox;
    [SerializeField] RectTransform itemBox;
    [SerializeField] RectTransform usingBox;

    [SerializeField] UIDescription description;
    [SerializeField] UIDragPanel dragPanel;

    int currentDragIndex = -1;
    UIItem currentAmmoItem;
    private void Awake()
    {
        Toggle(true);

        Init(equips, equipBox);
        Init(items, itemBox);
        Init(usings, usingBox);
        for (int i = 0; i < usings.Count; i++)
        {
            usings[i].InitUsingTxt(i + 1);
        }


        Toggle(false);
    }
    private void Start()
    {
        LoadData();
        CurrentEquipEffectsCheck?.Invoke();
    }
    public void SaveData()
    {
        JsonSaveLoader.Inventory_Save(equips, items, usings);

    }
    void LoadData()
    {
        InventoryData loadData = JsonSaveLoader.Inventory_Load();
        StatueController.Statue_StatsLoad();
        if(loadData == null)
        {

            return;
        }
        else
        {
            foreach (var item in loadData.values)
            {
                if (item.index == 0)
                {
                    equips[item.key].SetData(ItemManager.instance.GetItemByItemID(item.itemID),
item.quantity);
                }
                else if (item.index == 1)
                {
                    items[item.key].SetData(ItemManager.instance.GetItemByItemID(item.itemID),
item.quantity);
                }
                else if (item.index == 2)
                {
                    usings[item.key].SetData(ItemManager.instance.GetItemByItemID(item.itemID),
item.quantity);
                }
            }
        }

    }
    void Init(List<UIItem> items, RectTransform box)
    {
        items.Capacity = box.childCount;
        for (int i = 0; i < items.Capacity; i++)
        {
            items.Add(box.GetChild(i).GetComponent<UIItem>());
            items[i].ResetData();
            items[i].PointerRightClick += OnRightClick;
            items[i].PointerShiftRightClick += OnShiftRigthClick;
            items[i].PointerMove += DescriptionShow;
            items[i].PointerExit += DescriptionHide;
            items[i].BeginDrag += BeginDrag;
            items[i].EndDrag += EndDrag;
            items[i].Drop += Drop;
        }
    }

    void OnShiftRigthClick(UIItem uiItem)
    {
        if (Shop.IsShopOn)
        {
            Debug.Log("판매");
            uiItem.SellItem();
            SoundManager.instance.PlaySound(SoundType.SellItem);
        }

    }
    void OnRightClick(UIItem uiItem)
    {
        int index = items.IndexOf(uiItem);
        switch (uiItem.item.Type)
        {
            case ItemSO.TypeEnum.Helmet:
                ItemSwap(items, equips, index, 0);
                break;
            case ItemSO.TypeEnum.Armor:
                ItemSwap(items, equips, index, 1);
                break;
            case ItemSO.TypeEnum.Boots:
                ItemSwap(items, equips, index, 2);
                break;
            case ItemSO.TypeEnum.Accessories:
                ItemSwap(items, equips, index, 3);
                break;
            case ItemSO.TypeEnum.Consumable:
                UseItemAction?.Invoke(uiItem);
                uiItem.Updatequantity(-1);
                break;
        }

    }



    UIDescription DescriptionShow(UIItem uiItem)
    {
        ItemSO item = uiItem.item;
        description.SetData(description.Desc(item));
        return description;
    }
    void DescriptionHide(UIItem uiItem)
    {
        description.Toggle(false);
    }

    private void BeginDrag(UIItem uiItem)
    {
        int index = GetList(uiItem).IndexOf(uiItem);
        currentDragIndex = index;
        tmpList = GetList(uiItem);
        dragPanel.SetData(uiItem.item.itemImage, uiItem.Quantity);
        dragPanel.Toggle(true);
    }
    private void Drop(UIItem uiItem)
    {
        ItemSwap(tmpList, GetList(uiItem), currentDragIndex, GetList(uiItem).IndexOf(uiItem));
        currentDragIndex = -1;
    }


    public List<UIItem> GetList(UIItem uiItem)
    {
        if (equips.Contains(uiItem))
            return equips;
        else if (items.Contains(uiItem))
            return items;
        else if (usings.Contains(uiItem))
            return usings;
        else
            return null;
    }

    public UIItem InventoryIngredientCheck(ItemSO selectedItem)
```

```
{

    int ingredientID = selectedItem.ingredient[0].ingredient.ID;

    foreach (var item in items)
    {
        if (item.item == null)
            continue;
        else if (item.item.ID == ingredientID)
            return item;

    }
    foreach (var item in usings)
    {
        if (item.item == null)
            continue;
        else if (item.item.ID == ingredientID)
            return item;
    }
    Debug.Log("nullllllll");
    return null;
}

private void EndDrag(UIItem uiItem)
{
    dragPanel.Toggle(false);
}

UIItem HasPotionCheck()
{

    for (int i = 0; i < items.Count; i++)
    {
        if (items[i].item != null && items[i].item.Type == ItemSO.TypeEnum.Consumable)
        {
            Debug.Log("포션찾음 저장 : " + items[i].item.name);
            return items[i];
        }
    }


    return null;
}

public void PotionConsume()
{
    UseItemAction?.Invoke(HasPotionCheck());
    }

    public UIItem AmmoItemCheck()
    {

        if (currentAmmoItem == null)
        {
            for (int i = 0; i < items.Count; i++)
            {
                if (items[i].item != null)
                {
                    if (items[i].item.Type == ItemSO.TypeEnum.Ammo)
                    {
                        currentAmmoItem = items[i];
                    }
                }

                Debug.Log("포문도는거 체크");
            }
        }

        return currentAmmoItem;
    }

    public void UseAmmo(UIItem ammoItem)
    {
        ammoItem.Updatequantity(-1);
    }
    public void ItemSwap(List<UIItem> item1, List<UIItem> item2, int index1, int index2)
    {
        while (item1[index1].item.Type == ItemSO.TypeEnum.Accessories &&
!item2[index2].empty)
        {
            index2 += 1;
            if (index2 >= item2.Count)
            {
                index2 = 3;
                break;
            }
        }

        if (!item2[index2].empty)
        {

            ItemSO tmpItem = item2[index2].item;
            int tmpQuantity = item2[index2].Quantity;
            item2[index2].SetData(item1[index1].item.itemImage, item1[index1].Quantity,
item1[index1].item);
            item1[index1].SetData(tmpItem.itemImage, tmpQuantity, tmpItem);

        }
        else
        {
            item2[index2].SetData(item1[index1].item.itemImage, item1[index1].Quantity,
item1[index1].item);
            item1[index1].ResetData();
        }

        for (int i = 0; i < equips.Count; i++)
        {
            equips[i].ToggleQuantityTxt(false);
        }
        CurrentEquipEffectsCheck?.Invoke();
    }

    public void Toggle(bool val)
    {
        gameObject.GetComponent<VerticalLayoutGroup>().padding.top = val == true ? -150 :
0;
        equipBox.parent.gameObject.SetActive(val);
        itemBox.parent.gameObject.SetActive(val);
        currentAmmoItem = null;
    }

    public void GetItem(ItemSO item,int quantity=1)
    {
        SoundManager.instance.PlaySound(SoundType.GetItem);
        if (item.name == "Money")
        {
            GameManager.instance.Money += quantity;
            return;
        }


        //이미잇는거에서 수량추가
        for (int i = 0; i < items.Count; i++)
        {
            if (items[i].item != null)
            {
                if (items[i].item.ID == item.ID && item.IsStackable)
                {

                    items[i].Updatequantity(quantity);
                    return;
                }
            }

        }

        for (int i = 0; i < usings.Count; i++)
        {
            if (usings[i].item != null)
            {
                if (usings[i].item.ID == item.ID && item.IsStackable)
                {
                    usings[i].Updatequantity(quantity);
                    return;
                }
            }
        }
        for (int i = 0; i < items.Count; i++)
        {

            if (items[i].empty)
            {
                items[i].SetData(item.itemImage, quantity, item);
                return;
            }


        }

    }

    public ItemSO ItemSelected(int index)
    {
        for (int i = 0; i < usings.Count; i++)
        {
            usings[i].ItemSelected(false);
        }
        //index = index < 0 ? -index : index;
        //index = index % usings.Count;

        index = (int)Mathf.Repeat(index, 6);
        usings[index].ItemSelected(true);

        return usings[index].item;
    }
}


public class UIDescription : MonoBehaviour
```

```
{
    [SerializeField] TMP_Text titleTxt;

    private void Awake()
    {
        Toggle(false);
    }

    public void Toggle(bool val)
    {
        gameObject.SetActive(val);
    }

    public void SetData(StringBuilder stringBuilder)
    {
        titleTxt.text = stringBuilder.ToString();
        gameObject.SetActive(true);
    }
    public StringBuilder Desc(ItemSO item)
    {
        StringBuilder desc = new StringBuilder();
        desc.AppendLine(item.Name);
        desc.AppendLine(item.Type.ToString());
        desc.AppendLine(item.Description);
        EffectSO[] effects = item.effects;
        WeaponSO weaponItem = item as WeaponSO;
        ProjectileSO projectile = item as ProjectileSO;
        if (weaponItem != null)
        {
            AppendLineIfNotZero(desc, "Weapon Type : ", weaponItem.weaponType.ToString());
            AppendLineIfNotZero(desc, "Damage : ", weaponItem.damage);
            AppendLineIfNotZero(desc, "AttackSpeed : ", weaponItem.attackRate);
            AppendLineIfNotZero(desc, "Critical Chance : ", weaponItem.criticalChance + "%");
        }
        if(projectile != null)
        {
            AppendLineIfNotZero(desc, "Ammo Damage : ", projectile.damage);
        }
        if(effects.Length > 0)
        {
            AppendLineIfNotZero(desc, "Recovery Health : ", effects[0].CureHealthAmount);
            AppendLineIfNotZero(desc, "Add Health : ", effects[0].AddHealthAmount);
            AppendLineIfNotZero(desc, "Add Damage : ", effects[0].DamageAmount);
            AppendLineIfNotZero(desc, "Add Defense : ", effects[0].DefenseAmount);
        }

        return desc;
    }
    void AppendLineIfNotZero(StringBuilder sb, string restString, float? value)
    {
        if (value != 0)
        {
            sb.AppendLine(restString +value.ToString());
        }
    }
    void AppendLineIfNotZero(StringBuilder sb, string restString, string value)
    {
        if (value != null)
        {
            sb.AppendLine(restString + value);
        }
    }
}

public class InventoryController : MonoBehaviour
{
    [SerializeField] UIInventory inventory;
    WeaponController weaponController;
    int itemIndex;
    bool toggle;
    float curPotionCoolTime;
    float basePotionCoolTime =45f;
    float PotionCoolTime { get; set; } //아직 안씀
    float autoSaveTimer;
    float autoSaveTimerCycle = 300f;
    [SerializeField] Image potionCoolTimeImage;
    private void Start()
    {
        weaponController = GetComponent<WeaponController>();
        inventory.UseItemAction += UesItem;
        inventory.CurrentEquipEffectsCheck += CurrentEquipEffects;
        PotionCoolTime += basePotionCoolTime;
        curPotionCoolTime = PotionCoolTime;
        CurrentEquipEffects();
    }
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.I))
        {
            toggle = !toggle;
            inventory.Toggle(toggle);
        }

        ItemChange();
    Potion();

    AutoSave();
}

private void Potion()
{
    if (curPotionCoolTime <= PotionCoolTime)
    {
        curPotionCoolTime += Time.deltaTime;
        potionCoolTimeImage.fillAmount = 1 - curPotionCoolTime / PotionCoolTime;
    }
    else
    {
        potionCoolTimeImage.transform.parent.gameObject.SetActive(false);
    }
    if (Input.GetKeyDown(KeyCode.H) && curPotionCoolTime >= PotionCoolTime)
    {
        potionCoolTimeImage.transform.parent.gameObject.SetActive(true);
        inventory.PotionConsume();
        SoundManager.instance.PlaySound(SoundType.UsePotion);
        curPotionCoolTime = 0;
    }
}

void AutoSave()
{
    autoSaveTimer += Time.deltaTime;
    if (autoSaveTimer >= autoSaveTimerCycle)
    {

        autoSaveTimer = 0;
    }
}
public void SaveData()
{
    inventory.SaveData();
}
private void ItemChange()
{
    if (!toggle)
    {
        if (Input.GetKeyDown(GetKeyPressed1to6()))
        {
            itemIndex = (int)GetKeyPressed1to6() - 49;
            weaponController.ChangeWeapon(inventory, itemIndex);
        }
```

```
            if (Input.GetAxis("Mouse ScrollWheel") > 0f)
            {
                weaponController.ChangeWeapon(inventory, ++itemIndex);
            }
            else if (Input.GetAxis("Mouse ScrollWheel") < 0f)
            {
                weaponController.ChangeWeapon(inventory, --itemIndex);
            }
        }

    }

    public void CurrentEquipEffects()
    {
        Debug.Log("착용중인아이템체크후 업데이트");
        gameObject.GetComponent<PlayerController>().CurrentEquipEffects(inventory.equips);
    }


    public void GetItem(ItemSO item)
    {
        inventory.GetItem(item);
    }

    public UIItem CraftIngredientCheck(ItemSO item)
    {
        return inventory.InventoryIngredientCheck(item);
    }

    public void UesItem(UIItem UIItem)
    {
        UIItem.item.effects[0]?.Apply(gameObject.GetComponent<PlayerController>());
        UIItem.Updatequantity(-1);
    }

    public UIItem AmmoItemCheck()
    {
        return inventory.AmmoItemCheck();
    }
    public void UseAmmo(UIItem ammoItem)
    {
        inventory.UseAmmo(ammoItem);
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Item"))
        {
            Item item = collision.GetComponent<Item>();
            inventory.GetItem(item.item,item.Quantity);

            collision.gameObject.SetActive(false);
        }
    }


    private KeyCode GetKeyPressed1to6()
    {
        if (Input.GetKeyDown(KeyCode.Alpha1)) return KeyCode.Alpha1;
        if (Input.GetKeyDown(KeyCode.Alpha2)) return KeyCode.Alpha2;
        if (Input.GetKeyDown(KeyCode.Alpha3)) return KeyCode.Alpha3;
        if (Input.GetKeyDown(KeyCode.Alpha4)) return KeyCode.Alpha4;
        if (Input.GetKeyDown(KeyCode.Alpha5)) return KeyCode.Alpha5;
        if (Input.GetKeyDown(KeyCode.Alpha6)) return KeyCode.Alpha6;
        return KeyCode.None;
    }

    public void AddPotionCoolTime(float amount)
    {
        PotionCoolTime += amount;
    }
}

public class BlackSmithController : Shop
{
    [Header("Main")]
    [SerializeField] RectTransform shop;
    [SerializeField] RectTransform weaponCraft;
    [SerializeField] Button shopBtn;
    [SerializeField] Button craftBtn;

    [Space(20)]
    [Header("Craft")]
    [SerializeField]Transform craftTable;
    [SerializeField] Image craftImage;
    [SerializeField] Image craftIngredient1_Image;
    [SerializeField] Image craftIngredient2_Image;
    [SerializeField] TMP_Text craftIngredient1_Txt;
    [SerializeField] TMP_Text craftIngredient2_Txt;
    [SerializeField] Button itemCraftBtn;

    ItemSO getItem;

    ScrollRect scrollRect;

    List<UIShopItem> crafts = new List<UIShopItem>();
```

```
new void Awake()
{
    base.Awake();
    scrollRect = shopUI.GetComponent<ScrollRect>();
    Init(crafts, weaponCraft,CraftTable);

    shopBtn.onClick.AddListener(() =>
    {
        shop.gameObject.SetActive(true);
        weaponCraft.gameObject.SetActive(false);
        craftTable.gameObject.SetActive(false);
        scrollRect.content = shop;
    });
    craftBtn.onClick.AddListener(() =>
    {
        weaponCraft.gameObject.SetActive(true);
        shop.gameObject.SetActive(false);
        craftTable.gameObject.SetActive(false);
        scrollRect.content = weaponCraft;
    });

    itemCraftBtn.onClick.AddListener(() =>
    {
        itemCraftBtn.interactable = false;
        Craft();
    });

}

public void CraftTable(UIShopItem shopItem)
{

    int index = crafts.IndexOf(shopItem);
    craftTable.gameObject.SetActive(true);
    ShowInfo(crafts[index].itemSO); //클릭된 아이템의 정보를 가져옴

}

public void ShowInfo(ItemSO currentItem)
{

    UIItem currentInventoryItem = InventoryIngredientCheck(currentItem); // 해당무기의
재료로쓰는 인벤토리의 아이템을 받아옴
    int currentQuantity;
    if (currentInventoryItem == null)
        currentQuantity = 0;
    else
        currentQuantity = currentInventoryItem.Quantity;
    getItem = currentItem;

    craftImage.sprite = currentItem.itemImage;
    craftIngredient1_Image.sprite = currentItem.ingredient[0].ingredient.itemImage;
    craftIngredient2_Image.sprite = currentItem.ingredient[1].ingredient.itemImage;

    craftIngredient1_Txt.color = IsIngredientEnough(currentQuantity,
currentItem.ingredient[0].count);
    craftIngredient2_Txt.color = IsIngredientEnough(GameManager.instance.Money,
currentItem.ingredient[1].count);

    craftIngredient1_Txt.text = currentQuantity + " / " + currentItem.ingredient[0].count;
    craftIngredient2_Txt.text =
MyUtils.GetThousandCommaText(GameManager.instance.Money) + " / " +
                                    MyUtils.GetThousandCommaText(currentItem.ingredient[1].count); //
소지중인금액 / 필요한금액

    if (craftIngredient1_Txt.color == Color.green && craftIngredient2_Txt.color ==
Color.green)
    {
        itemCraftBtn.interactable = true;
    }
    else
        itemCraftBtn.interactable = false;
}

Color IsIngredientEnough(int currentQuantity, int neededQuantity)
{
    if (currentQuantity < neededQuantity)
        return Color.red;
    else
        return Color.green;
}

public UIItem InventoryIngredientCheck(ItemSO item)
{

    return inventory?.CraftIngredientCheck(item);

}

public void Craft()
{
    UIItem currentInventoryItem = InventoryIngredientCheck(this.getItem); // 해당무기의
재료로쓰는 인벤토리의 아이템을 받아옴

    currentInventoryItem.Quantity -= getItem.ingredient[0].count;
    GameManager.instance.Money -= getItem.ingredient[1].count;

    currentInventoryItem.Updatequantity(-getItem.ingredient[0].count);
    inventory.GetItem(this.getItem);
    ShowInfo(this.getItem);
}
}

public class Shop : Interact
{
protected List<UIShopItem> items = new List<UIShopItem>();
[SerializeField] Transform slotTransform;
public static bool IsShopOn { get; private set; } = false;
protected void Awake()
{
    Init(items, slotTransform,ItemBuy);
    CheckShop += CheckShopOn;
}
public void CheckShopOn()
{
    IsShopOn = shopUI.activeSelf;
}
public void Init(List<UIShopItem> items, Transform slot, Action<UIShopItem> action)
{
    for (int i = 0; i < slot.childCount; i++)
    {
        items.Add(slot.GetChild(i).GetComponent<UIShopItem>());
        items[i].ItemClicked += action;
    }
}

public void ItemBuy(UIShopItem shopItem)
{
    if (GameManager.instance.Money >= shopItem.itemSO.BuyCost)
    {
        GameManager.instance.Money -= shopItem.itemSO.BuyCost;
        inventory.GetItem(shopItem.BuyItem());
        SoundManager.instance.PlaySound(SoundType.BuyItem);
    }
    else
    {
        SoundManager.instance.PlaySound(SoundType.NotEnoughMoney);
        Debug.Log("돈이부족합니다. :" + GameManager.instance.Money);
    }

}
}
```

```
public interface IInteractable
{
   public void interact();
}
public class Interact : MonoBehaviour, IInteractable
{
   bool isIn = false;
   public bool Toggle { get; private set; } = false;
   public event Action CheckShop;
   [SerializeField]
   protected GameObject shopUI;
   GameObject interactUI;
   protected InventoryController inventory;

   protected float colliderRadius = 3.5f;
   private void Start()
   {
      interactUI = shopUI.transform.parent.gameObject;
      CircleCollider2D circleCollider2D = gameObject.AddComponent<CircleCollider2D>();
      circleCollider2D.radius = colliderRadius;
      circleCollider2D.isTrigger = true;

   }
   public void OnTriggerEnter2D(Collider2D collision)
   {
      if (collision.CompareTag("Player"))
      {
         inventory = collision.GetComponent<InventoryController>();
         isIn = true;
      }
   }
   public void OnTriggerExit2D(Collider2D collision)
   {
      if (collision.CompareTag("Player"))
      {
         isIn = false;
         if (shopUI.activeSelf)
         {
            objectToggle(false);
            Debug.Log("OnTriggerExit2D : 꺼짐");

         }
      }

   }

   public virtual void interact()
   {
        objectToggle(isIn);
    }
    void objectToggle(bool val)
    {
        shopUI.SetActive(val);
        interactUI.SetActive(val);
        Toggle = val;
        CheckShop?.Invoke();

    }
}
```