```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using uikanri;

public class UiManager : MonoBehaviour
{
    //card manager をCardManager スクリプトに参照
    public Card card;
    public CardManager cardManager;

    //
    public GameObject[] cardSlots;
    public GameObject cardDisplaySlots;
    public GameObject[] Decktext;


    //ページの変数
    public int page;
    public Text pageText;

    public int cnt1;
    public int cnt2;


    //初期化処
    public void Start()
    {
        //カード一覧表示
        DisplayCards();

        //カード拡大表示
        CardDisplaySlots();

        //参照したデッキケースの内容をロード
        PlayerPrefs.GetInt("save");

        for (int i = 0; i < cardSlots.Length; i++)
        {
        cardSlots[i].GetComponent<UiUser>().cardID = i;
        }
    }

    //更新
    public void Update()
    {
        //ページの処理
        UpdatePage();

        if (Input.GetKeyDown(KeyCode.D))//Dキーで次のページへ進む
        {
            if (page >= Mathf.Floor((cardManager.cards.Count - 1) / 4))
            {
                page = 0;
            }
            //ページの値を増やす
            else
            {
                page++;
            }
            Debug.Log("ーー");
        }

        if (Input.GetKeyDown(KeyCode.A))//Aキー前のページに戻る
        {
            if (page <= 0)
            {
                page = Mathf.Floor.ToInt((cardManager.cards.Count - 1) / 4);
            }
            //ページの値を減らす
            else
            {
                page--;
            }
            Debug.Log("ーー");
        }

        DisplayCards();//表示させる場所はここですという意味


        if (cnt1-- == 0)
        {
            Decktext[0].gameObject.SetActive(false);
        }
        if(cnt2-- == 0)
        {
            Decktext[1].gameObject.SetActive(false);
        }


    }

    //ページの変更
    private void UpdatePage()
    {
        //ページテキストを文字型にする。　例）3　/8は/カードごとに数を変えてもいい意味
        //pageTextInt = (page + 1).ToString();
        //pageTextInt = (page + 1) + "/" + (Mathf.Ceil(cardSlots.Length / 8) + 1).ToString();
        //MathfCeil 切り上げ
        pageTextInt = (page + 1) + "/" + (Mathf.Ceil(cardSlots.Length / 4.0f) + "ページ").ToString();

    }

    //カード表示
    public void DisplayCards()
    {
        // cardManager の cards の情報を各 cardSlots に割り当て ↓
        for (int i = 0; i < cardSlots.Length; i++)
        {
            //0) > 0 && i < 8)
            // 1 ページ
            //0) > 8 && i < 16 )
            //2ページ
            //0) > 16 && i < 24)
            //3ページ
            //ページについて　カードスロットの数よりも上のページにいき、ページの値を増やしたらこの後に続くものを表示
            if( i >= page * 4 && i < (page + 1) * 4)
            {
                cardSlots[i].gameObject.SetActive(true);

                //cardSlots[i]の(カード番)　に対応した名前を表示。
                //cardSlots[i]の中の配列2番目にある（name)image()の中の配列0番目にある(name(s)）を参照して名前テキストを表示
                cardSlots[i].transform.GetChild(1).transform.GetChild(0).GetComponent<Text>().text = cardManager.cards[i].CardName;

                //cardSlots[i]の中の配列2番目にある（CardBody)の中の配列1番目にある（cardDes)を参照して説明テキストを表示


                //cardSlots[i]の中の配列　番目にある（）の中の配列　番目にある（）を参照してアノーンフンを表示
                cardSlots[i].transform.GetChild(2).transform.GetChild(0).GetComponent<Image>().sprite = cardManager.cards[i].JankenSprite;

                //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列2番目にある（kimou）を参照して敵物画像を表示
                cardSlots[i].transform.GetChild(2).transform.GetChild(2).GetComponent<Image>().sprite = cardManager.cards[i].CardSprite;
                //cardSlots[i]の中の配列0番目にある（CardBack）を参照して背面画像を表示
                cardSlots[i].transform.GetChild(0).GetComponent<Image>().sprite = cardManager.cards[i].Back;

                //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列3番目にある（Attack）を参照して攻撃力テキストを表示
                cardSlots[i].transform.GetChild(2).transform.GetChild(3).GetComponent<Text>().text = cardManager.cards[i].CardAttack.ToString();
                //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列4番目にある（Defence）を参照して防御力テキストを表示
                cardSlots[i].transform.GetChild(2).transform.GetChild(4).GetComponent<Text>().text = cardManager.cards[i].CardDefence.ToString();

                cardSlots[i].transform.GetChild(2).GetComponent<Image>().sprite = cardManager.cards[i].Rarity;


                //カードのロック中用カード表示
                #region New Features


                //Card中の変数 isHeld!!　falseの時
                if (cardManager.cards[i].isHeld == false)
                {
                    //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列5番目にある(Lock)を表示する。
                    cardSlots[i].transform.GetChild(2).transform.GetChild(5).gameObject.SetActive(true);
                    //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列2番目にある(kimou)を表示する。
                    cardSlots[i].transform.GetChild(2).transform.GetChild(2).GetComponent<Image>().color = new Color(0, 0, 0);
                    //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列6番目にある(kimou) image(敵画)　を表示しない。

                }
                else
                {
                    //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列5番目にある(Lock)を表示しない。
                    cardSlots[i].transform.GetChild(2).transform.GetChild(5).gameObject.SetActive(false);
                    //cardSlots[i]の中の配列2番目にある（CardBody）の中の配列2番目にある(kimou)を表示する。
                    cardSlots[i].transform.GetChild(2).transform.GetChild(2).GetComponent<Image>().color = new Color(255, 255, 255);


                    if (cardManager.cards[i].holdNumber >= 0)
                    {
                        //*カード枚数を表示
                        cardSlots[i].transform.GetChild(2).transform.GetChild(6).transform.GetChild(0).GetComponent<Text>().text = "x " + cardManager.cards[i].holdNumber.ToString();

                    }

                }
                #endregion
            }
            else
            {
                //カードを表示にする。
                cardSlots[i].gameObject.SetActive(false);
            }
        }
    }

    //カード拡大表示
    public void CardDisplaySlots()
    {
        cardDisplaySlots.gameObject.SetActive(false);
    }


    public void DeckMenperText()
    {
        Decktext[0].gameObject.SetActive(true);
        cnt1 = 200;
    }

    public void DeckSaveText()
    {
        Decktext[1].gameObject.SetActive(true);
        cnt2 = 200;
    }

    //カードデータの表示
    public void cardDisplaySetUp ()
        {


            Debug.Log("OnClickCardBtnButton cardManager.cards[" + i + "].holdNumber = " + cardManager.cards[i].holdNumber.ToString());
            //Log出力されたものを表示する。
            CardDisplaySlots();
            //カード表示
            cardDisplaySlots.gameObject.SetActive(true);
            //cardDisplaySlots[i]の中の配列2番目にある（name)の中の配列0番目にある（name)を参照して説明テキストを表示
            cardDisplaySlots.transform.GetChild(1).transform.GetChild(0).GetComponent<Text>().text = cardManager.cards[i].CardName;

            cardDisplaySlots.transform.GetChild(2).transform.GetChild(0).GetComponent<Image>().sprite = cardManager.cards[i].JankenSprite;

            //cardDisplaySlots[i]の中の配列2番目にある（CardBody）の中の配列2番目にある（kimou）を参照して敵物画像を表示
            cardDisplaySlots.transform.GetChild(2).transform.GetChild(2).GetComponent<Image>().sprite = cardManager.cards[i].CardSprite;
            //cardDisplaySlots[i]の中の配列0番目にある（CardBack）を参照して背面画像を表示
            cardDisplaySlots.transform.GetChild(0).GetComponent<Image>().sprite = cardManager.cards[i].Back;

            //cardDisplaySlots[i]の中の配列2番目にある（CardBody）の中の配列3番目にある（Attack）を参照して攻撃力テキストを表示
            cardDisplaySlots.transform.GetChild(2).transform.GetChild(3).GetComponent<Text>().text = cardManager.cards[i].CardAttack.ToString();
            //cardDisplaySlots[i]の中の配列2番目にある（CardBody）の中の配列4番目にある（Defence）を参照して防御力テキストを表示
            cardDisplaySlots.transform.GetChild(2).transform.GetChild(4).GetComponent<Text>().text = cardManager.cards[i].CardDefence.ToString();
            //cardDisplaySlots[i]の中の配列2番目にある（CardBody）を参照して表示する。
            cardDisplaySlots.transform.GetChild(2).transform.GetChild(5).gameObject.SetActive(false);
            //cardDisplaySlots[i]の中の配列2番目にある（CardBody）の中の配列6番目にある（holdNumber）を参照して表示にする。
            cardDisplaySlots.transform.GetChild(2).transform.GetChild(6).gameObject.SetActive(false);
        }

}
```