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


//インスタンス生成

public class CardDeck : MonoBehaviour
{
    //Card/CardManagerクラス参照
    public Card card;
    public CardManager cardManager;
    public UiManager uiManager;


    //カード情報表示させるためのslots
    public GameObject[] deckSlots;

    //配列要素６
    //デッキ枚数の上限６枚
    public int[] cardDeck = new int[6];

    // Start is called before the first frame update
    void Start()
    {
        //初期状態はデッキ非表示
        CardDeckDisplayFalse();



        //cardDeckを初期化
        CardDelete();
    }

    // Update is called once per frame
    void Update()
    {

    }
    public void CardDeckSave()
    {

    }

    public void CardcaseDelete()
    {

    }



    //デッキのカード表示メソッド
    public void CardDeckDisplay()
    {
        CardDeckDisplayFalse();

        for (int j = 0; j < cardDeck.Length; j++)
        {
            int i = cardDeck[j];

            if(cardDeck[j] == -1)
            {
                continue;
            }

            deckSlots[j].gameObject.SetActive(true);


            //deckSlots[i](カード事）に任意した名前を表示。
            //deckSlot(i)の中の配列1番目にある（name(image)）の中の配列0番目にある(name(tx))を参照して名前テキストを表示
            deckSlots[j].transform.GetChild(1).transform.GetChild(0).GetComponent<Text>().text = cardManager.cards[i].CardName;

            //deckSlot(i)の中の配列　番目にある（）の中の配列 番目にある（）を参照してジャンケンを表示
            deckSlots[j].transform.GetChild(2).transform.GetChild(0).GetComponent<Image>().sprite = cardManager.cards[i].JankenSprite;

            //deckSlot(i)の中の配列２番目にある（CardBody）の中の配列２番目にある（ikimono）を参照して動物画像を表示
            deckSlots[j].transform.GetChild(2).transform.GetChild(2).GetComponent<Image>().sprite = cardManager.cards[i].CardSprite;
            //cardSlot(i)の中の配列0番目にある（Cardhaikei)を参照して背景画像を表示
            deckSlots[j].transform.GetChild(0).GetComponent<Image>().sprite = cardManager.cards[i].Back;

            //deckSlot(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
            deckSlots[j].transform.GetChild(2).transform.GetChild(3).GetComponent<Text>().text = cardManager.cards[i].CardAttack.ToString();
            //deckSlot(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
            deckSlots[j].transform.GetChild(2).transform.GetChild(4).GetComponent<Text>().text = cardManager.cards[i].CardDefense.ToString();
            deckSlots[j].transform.GetChild(2).GetComponent<Image>().sprite = cardManager.cards[i].Rarity;



        }
    }

    //デッキカード非表示メソッド
    public void CardDeckDisplayFalse()
    {
        for (int i = 0; i < deckSlots.Length; i++)
        {
            deckSlots[i].gameObject.SetActive(false);
        }
    }

    //デッキ内にカードを格納するメソッド
    public void CardSet(int i)
    {
        for (int j = 0; cardDeck.Length > j; j++)
        {
            Debug.Log(cardDeck[j]);
            if (cardDeck[j] == i)
            {
                return;
            }
            if (cardDeck[j] == -1)
            {
                cardManager.cards[i].holdNumber -= 1;
                cardDeck[j] = i;
                return;
            }
        }
         Debug.Log("デッキがいっぱい");
        //↑の処理が終わったら（６枚以上になったら）テキスト・イメージを表示する
        uiManager.DeckManpaiText();

    }

    //デッキ初期化
    public void CardDelete()
    {
        for(int j = 0; cardDeck.Length > j; j++)
        {
            cardDeck[j] = -1;
        }
        Debug.Log("デッキを初期化しました");
    }
}
```