```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;

class Contact {
  String name;
  String phoneNumber;
  String email;
  String address;

  public Contact(String name, String phoneNumber, String email, String address) {
    this.name = name;
    this.phoneNumber = phoneNumber;
    this.email = email;
    this.address = address;
  }
}

class ContactManagementSystem {
  private static final int TABLE_SIZE = 10;
  private LinkedList<Contact>[] hashTable;
  private Stack<Contact> undoStack;
  private Stack<Contact> redoStack;
  private Queue<String> callLog;

  public ContactManagementSystem() {
    hashTable = new LinkedList[TABLE_SIZE];
    undoStack = new Stack<>();
    redoStack = new Stack<>();
    callLog = new LinkedList<>();

    for (int i = 0; i < TABLE_SIZE; i++) {
      hashTable[i] = new LinkedList<>();
    }
  }

  // Hash function to get the index in the hash table
  private int hashFunction(String key) {
    // Simplified hash function (for demonstration purposes)
    return key.length() % TABLE_SIZE;
  }

  // Add a new contact to the system
  public void addContact(String name, String phoneNumber, String email, String address) {
    Contact newContact = new Contact(name, phoneNumber, email, address);
    int index = hashFunction(name);

    // Handle collisions using linked lists
    hashTable[index].add(newContact);

    // Update undo stack
    undoStack.push(newContact);
    redoStack.clear(); // Clear redo stack after adding a new contact

    System.out.println("Contact added successfully!");
```

```
  }

  // Undo the previous contact modification
  public void undo() {
    if (!undoStack.isEmpty()) {
      Contact removedContact = undoStack.pop();
      int index = hashFunction(removedContact.name);
      hashTable[index].remove(removedContact);

      // Update redo stack
      redoStack.push(removedContact);

      System.out.println("Undo successful!");
    } else {
      System.out.println("Nothing to undo.");
    }
  }

  // Redo the previous undone contact modification
  public void redo() {
    if (!redoStack.isEmpty()) {
      Contact restoredContact = redoStack.pop();
      int index = hashFunction(restoredContact.name);
      hashTable[index].add(restoredContact);

      // Update undo stack
      undoStack.push(restoredContact);

      System.out.println("Redo successful!");
    } else {
      System.out.println("Nothing to redo.");
    }
  }

  // Display the call log
  public void displayCallLog() {
    System.out.println("Call Log:");
    for (String call : callLog) {
      System.out.println(call);
    }
  }

  // Make a call and add it to the call log
  public void makeCall(String callDetails) {
    callLog.add(callDetails);

    // Limit the call log size to 10 for demonstration purposes
    if (callLog.size() > 10) {
      callLog.poll(); // Remove the oldest entry
    }

    System.out.println("Call made successfully!");
  }
}
```

```
public class project_4 {

  public static void main(String[] args) {
    ContactManagementSystem cms = new ContactManagementSystem();

    // Add contacts
    cms.addContact("John Doe", "1234567890", "john.doe@email.com", "123 Main St");
    cms.addContact("Jane Smith", "9876543210", "jane.smith@email.com", "456 Oak St");

    // Undo a contact addition
    cms.undo();

    // Redo the undone contact addition
    cms.redo();

    // Make calls
    cms.makeCall("Call to John Doe");
    cms.makeCall("Call to Jane Smith");

    // Display the call log
    cms.displayCallLog();
  }
}
```