1. **Java Code:**
2. import java.util.Arrays;  
   class Kisi {  
    String isim;  
    String telefon;  
     
    public Kisi(String isim, String telefon) {  
    this.isim = isim;  
    this.telefon = telefon;  
    }  
   }  
     
   public class TelefonRehberi {  
    private static final int *MAX\_KISI\_SAYISI* = 20;  
    private Kisi[] rehber;  
     
    public TelefonRehberi() {  
    rehber = new Kisi[*MAX\_KISI\_SAYISI*];  
    }  
     
    public void ekle(String isim, String telefon) {  
    if (kayitSayisi() >= *MAX\_KISI\_SAYISI*) {  
    System.*out*.println("Rehber dolu, yeni kayıt eklenemiyor.");  
    return;  
    }  
     
    int hashIndex = hash(isim.charAt(0));  
    if (rehber[hashIndex] != null && rehber[hashIndex].isim.equals(isim)) {  
    System.*out*.println("Bu isim zaten rehberde var.");  
    return;  
    }  
     
    rehber[hashIndex] = new Kisi(isim, telefon);  
    System.*out*.println("Yeni kayıt eklendi: " + isim + ", " + telefon);  
    }  
     
    public void ara(String isim) {  
    int hashIndex = hash(isim.charAt(0));  
    if (rehber[hashIndex] == null || !rehber[hashIndex].isim.equals(isim)) {  
    System.*out*.println("Kişi bulunamadı: " + isim);  
    return;  
    }  
     
    System.*out*.println(isim + " kişisinin telefon numarası: " + rehber[hashIndex].telefon);  
    }  
     
    public void sil(String isim) {  
    int hashIndex = hash(isim.charAt(0));  
    if (rehber[hashIndex] == null || !rehber[hashIndex].isim.equals(isim)) {  
    System.*out*.println(isim + " kişisi bulunamadığı için silinemedi!");  
    return;  
    }  
     
    rehber[hashIndex] = null;  
    System.*out*.println(isim + " rehberden silindi.");  
    }  
     
    private int hash(char ch) {  
    return (int) ch % *MAX\_KISI\_SAYISI*;  
    }  
     
    private int kayitSayisi() {  
    return (int) Arrays.*stream*(rehber).filter(kayit -> kayit != null).count();  
    }  
     
    public static void main(String[] args) {  
    TelefonRehberi rehber = new TelefonRehberi();  
     
    rehber.ekle("Ayşe", "5554443322");  
    rehber.ekle("Ahmet", "5551234567");  
    rehber.ekle("Mehmet", "5559876543");  
     
    rehber.ara("Ahmet");  
    rehber.ara("Ali");  
     
    rehber.sil("Mehmet");  
    rehber.ara("Mehmet");  
     
    rehber.sil("Enes");  
    }  
   }

**2.Çakışma çözümlemesi için seçtiğiniz yöntemi belirtiniz ve neden bu yöntemi tercih ettiğinizi açıklayınız.**

Bu uygulamada basit bir hashing yöntemi kullanılmıştır. İsimlerin ilk karakterinin ASCII değeri alınarak, bu değer rehberin bir indisine dönüştürülmüştür. Bu, çakışma (collision) olasılığını azaltmak için basit ama etkili bir yöntemdir. Verilen durumda, isimlerin benzersiz olduğu ve rehberin maksimum 20 kişiyi barındırabileceği belirtilmiştir. Dolayısıyla, basit bir hashing yöntemi yeterlidir ve performans açısından uygun bir seçenektir.

1. **Her bir fonksiyon için algoritmanın çalışma süresini analiz ediniz.**

* Ekleme (Insert): Ekleme işlemi, verilen ismin ASCII değerine göre hash hesaplaması yapar ve bu hesaplanan indekse yeni kişiyi ekler. Bu işlem sabit zamanda (O(1)) gerçekleşir.
* Arama (Search): Arama işlemi, verilen ismin ASCII değerine göre hash hesaplaması yapar ve bu hesaplanan indeksteki kişiyi arar. Bu işlem de sabit zamanda (O(1)) gerçekleşir.
* Silme (Delete): Silme işlemi, verilen ismin ASCII değerine göre hash hesaplaması yapar, bu hesaplanan indeksteki kişiyi null olarak işaretler. Bu işlem de sabit zamanda (O(1)) gerçekleşir.