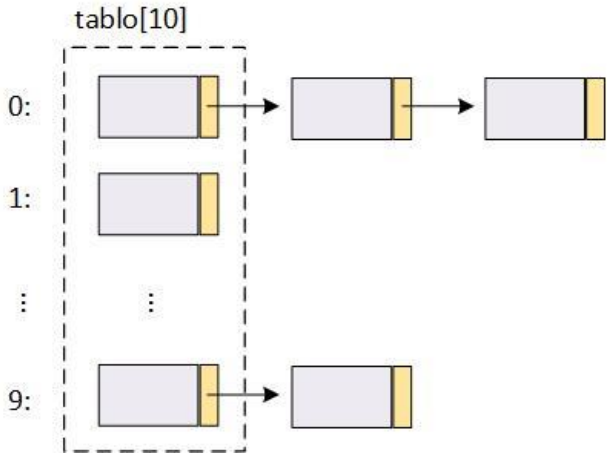
DOUBLE HASHING

Double hashing, index alanların çakışması durumunda i*hash2() kadar sonraki boş alanı arar ve gelen elemanı buraya ekler. Örneğin mod 10 a göre indexleme yapıldığını varsayalım. 53 ve 43 sayıları 3 indisine yazılmalıdır. Ancak 53'i ekledikten sonra 3. İndis dolu olduğundan 1. iterayonda i*hash2(53) sonrasına bakar. Has2(53) değeri 2 olduğu varsayılırsa bu sayı 3+2=5. İndise tekabül eder. 5. indis de dolu ise 2. İterasyonda (2*hash2(53)=4) sonrasına bakılır. Bu da 7. İndise denk gelmektedir. Boş yer arama işlemi bu şekilde devam eder. Sona ulaşması durumunda elde edilen indis değerinin tablo boyutuna göre modu alınır ve dairesel döngü oluşturulmuş olur. Ekleme, listeleme, arama ve silme işlemleri için gereken kodlar aşağıda verilmiştir. Quadratic Probing'ten farkı artis*artis yerine artis*hashFonksiyonu2(anahtar) sonrasına bakılmasıdır.

```
int hashFonksiyonu(int anahtar) {
    return anahtar % 11;
}

int hashFonksiyonu2(int anahtar) {
    return ((anahtar%7)*3) % 11;
}

void veriEkle(int anahtar) {
    int indis = hashFonksiyonu(anahtar);
    int yeniIndis = indis;
    int artis = 0;
    while(tablo[indis] != 0) {
        yeniIndis = (indis + artis*hashFonksiyonu2(anahtar))%11;
        artis ++;
    }
    tablo[yeniIndis] = anahtar;
}

int ara (int anahtar) {
    int indis = hashFonksiyonu(anahtar);
    if(anahtar == tablo[indis]) {
        return indis;
    }else {
        int yeniIndis = indis;
        int artis = 1;
        while(tablo[indis] != 0) {
            yeniIndis = (indis + artis*hashFonksiyonu2(anahtar)) % 11;
            if(anahtar == tablo[yeniIndis]) {
                return yeniIndis;
            }
            artis ++;
        }
    }
    return -1;
}
```

ZİNCİRLEME YÖNTEMİ

Zincirleme yönteminde çakışan elemanlar bağıl liste şeklinde tutulur. Bu sayede aynı indiste birden fazla eleman bulundurabiliriz. Şekil 1'de zincirleme yöntemi örneği verilmiştir.

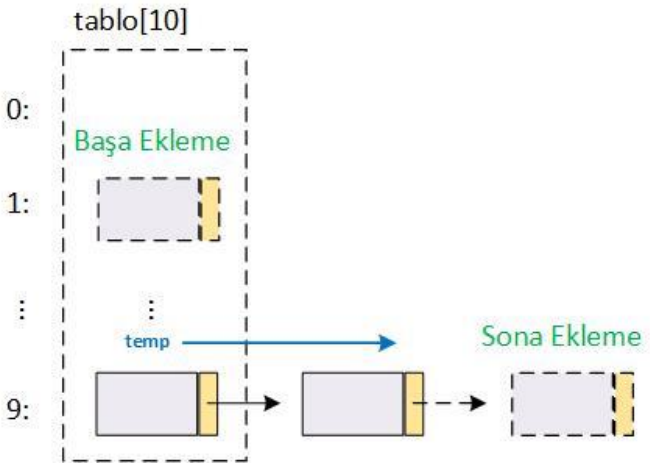


Şekil 1. Zincirleme yöntemi

Şekilden de görüleceği üzere tek yönlü bağıl liste ve düğüm yapısı kullanılmaktadır. Bu düğüm yapısı Java'da aşağıdaki şekilde tanımlanabilir.

```
public class Node {
    int veri;
    Node next;
}
```

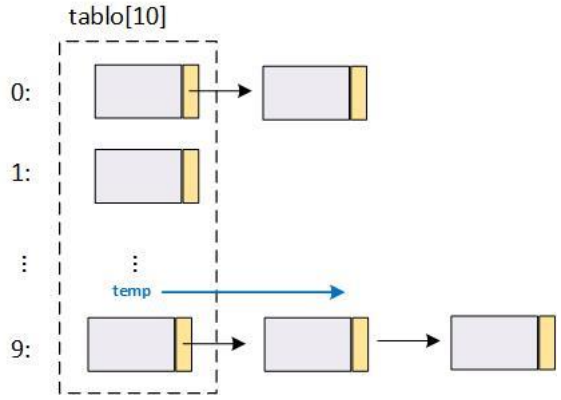
Bu şekilde tanımlanan bir tabloya eleman ekleme için öncelikle tablodaki ilgili indisin boş olup olmadığı kontrol edilir. Boş ise yeni oluşturulan Node tablonun ilgili indis değerine atanır. Değilse bağıl listenin sonuna gidilir ve Node sona eklenir. Ekleme işlemi şekil 2'deki gibi şekilde gerçekleştirilir.



Şekil 2. Zincirleme kuralı ile eleman ekleme

```
void ekle(int eleman) {
    Node yeni = new Node(eleman);
    int indis = hashFonksiyonu(eleman);

    if(tablo[indis] == null) {
        tablo[indis] = yeni;
    }else {
        Node temp = tablo[indis];
        while(temp.next != null) {
            temp = temp.next;
        }
        temp.next = yeni;
    }
}
```

Elemanlarda gezinmek için ilk elemanın sonrasından başlanır. Geçici değişkenimiz null olduğu zaman döngü sonlandırılır. Zincirleme yönteminde gezinme işlemi şekil 3 ile verilmiştir.

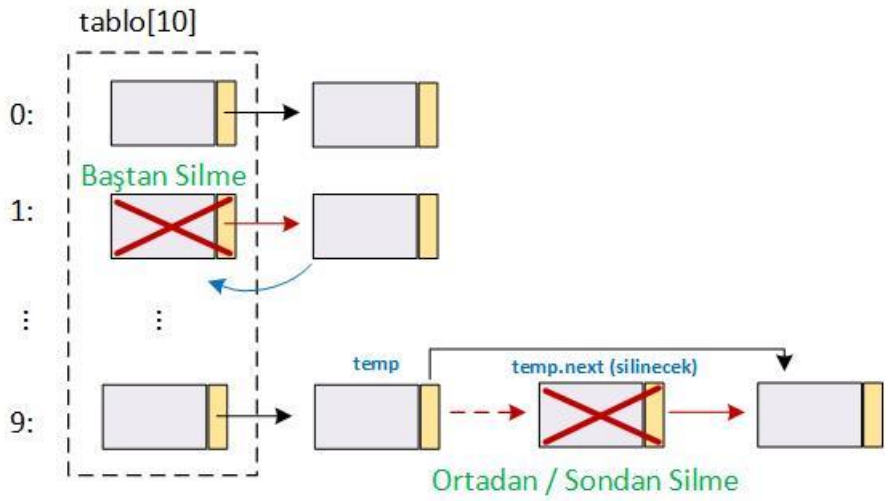


Şekil 3. Zincirleme yönteminde gezinme

Şekil 3'teki mantıkla listeleme ve arama işlemleri gerçekleştirilir. Listeleme ve arama kodları aşağıda verilmiştir.

```
void listele() {
    for (int i = 0; i < 10; i++) {
        System.out.println("["+i+"]:");
        Node temp = tablo[i];
        while(temp != null) {
            System.out.print(temp.veri + " ");
            temp = temp.next;
        }
        System.out.println("");
    }
}

Node ara(int eleman) {
    int indis = hashFonksiyonu(eleman);
    Node temp = tablo[indis];
    while(temp!=null) {
        if(temp.veri == eleman) {
            return temp;
        }
        temp = temp.next;
    }
    return null;
}
```

Baştan silme işlemi için tablonun ilgili indisi null yapılmalıdır. Ortadan veya sondan silme işlemi için ise silinecek düğümün gerisindeki düğümün nexti, silinecek elemanın nextine eşitlenmelidir. Bu sayede şekil 4'teki gibi kırmızı ile gösterilen bağlar koparılır. Silme işleminin kodu aşağıda verilmiştir.



Şekil 4. Zincirleme yöntemi silme işlemi

```
void sil(int eleman) {
    int indis = hashFonksiyonu(eleman);

    if(tablo[indis].veri == eleman) {
        tablo[indis] = tablo[indis].next;
    }else {
        Node temp = tablo[indis];
        while(temp.next != null) {
            if(temp.next.veri == eleman) {
                temp.next = temp.next.next;
            }
            temp = temp.next;
        }
    }
}
```