ArrayList

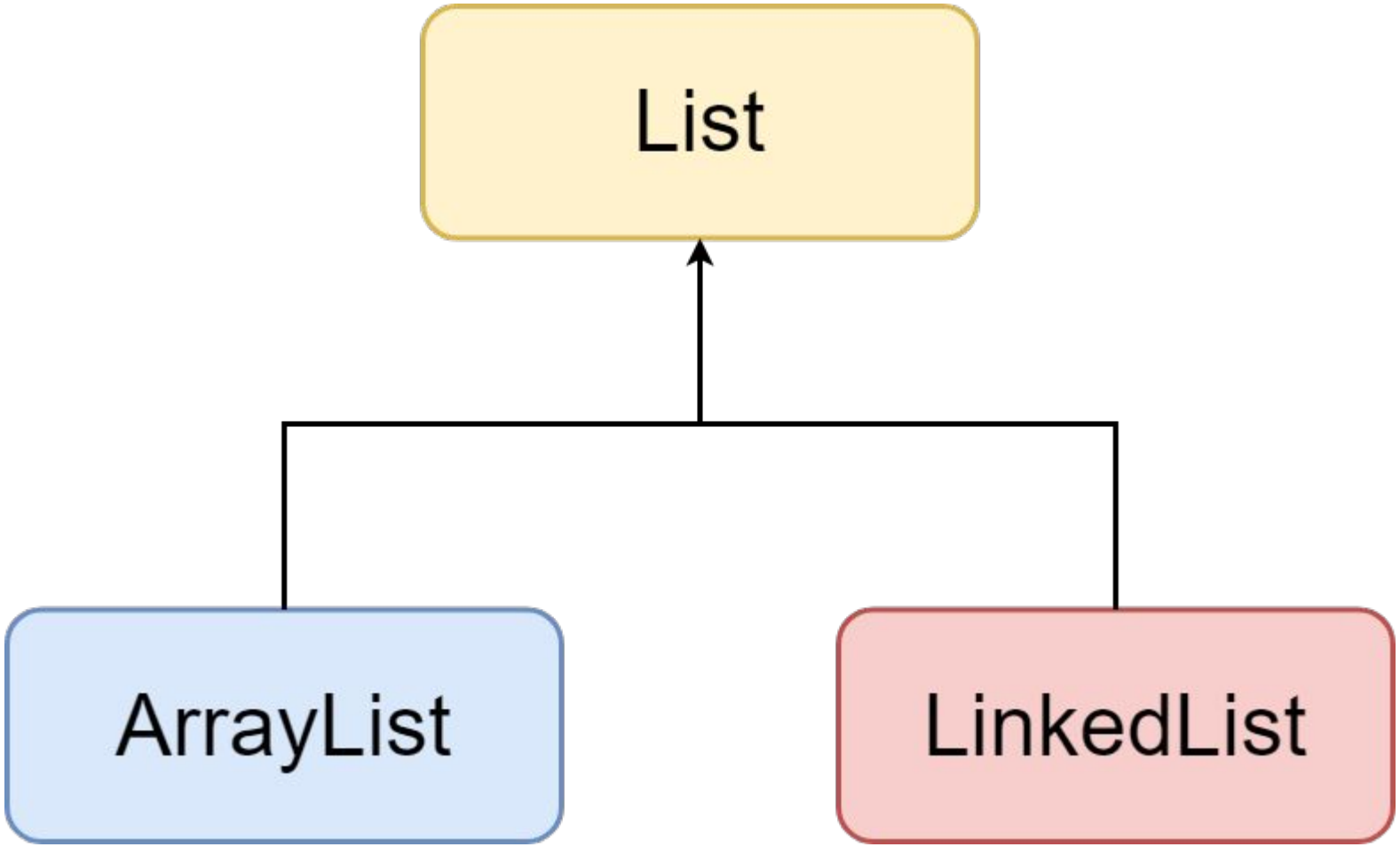
Algoritma & Struktur Data (SIF203)

Program Studi Informatika - Program Sarjana
Fakultas Teknologi Industri

List

- **List** merupakan sebuah struktur data yang dapat digunakan untuk ***menyimpan beberapa data*** dalam ***waktu yang bersamaan***.
- **Isian** (elemen) di dalam **List** memiliki sifat **dapat diubah (changeable)** kapan saja dan menempati urutan tertentu **(ordered)**
  - Terurut dalam artian setiap isian dalam **List** memiliki nomor urut yang teratur/pasti
  - Nomor urut disebut dengan nama **index** (dimulai dari **0** sebagai index pertama)
- Elemen **List** dapat dipanggil/ditampilkan berdasarkan nomor **index**.
- Sebuah **List** dapat memiliki elemen yang sama pada index yang berbeda
- Ukuran **List** memiliki panjang yang dapat berubah-ubah sehingga ***menambahkan*** dan ***menghapus*** elemen tertentu di dalam **List**.

Implementasi List



UNIVERSITAS ISLAM INDONESIA

ArrayList

ArrayList

- **ArrayList** adalah bentuk lain dari ***array*** yang memiliki kelebihan yaitu ukurannya yang dapat ***dimodifikasi***
- ***Array*** dalam Java ukurannya **tidak** dapat diubah (jika ingin menambah atau mengurangi elemen ***array***, kita harus **membuat *array*** yang **baru**)
- Jangkauan Array untuk menyimpan data harus dideklarasikan menggunakan elemen dalam ukuran maksimalnya sehingga Array tidak dapat digunakan jika diinginkan untuk menyimpan banyak data yang melebihi ukuran maksimal tersebut.
- Pada ***ArrayList*** kita dapat menambahkan atau menghapus elemen di manapun letaknya.

ArrayList

Beberapa operasi dapat Anda lakukan terhadap ArrayList seperti berikut:

- **isEmpty()**, untuk memeriksa apakah ArrayList kosong atau tidak
- **size()**, untuk mencari panjang ArrayList
- **get()**, untuk mengambil elemen pada index tertentu
- **indexOf()**, untuk mengetahui index dari suatu nilai (jika nilai tidak ada dalam ArrayList maka hasilnya adalah index -1)
- **contains()**, untuk memeriksa apakah suatu nilai ada dalam ArrayList
- **set()**, untuk mengganti nilai pada index tertentu
- **clear()**, menghapus semua elemen ArrayList
- **add()**, untuk menambah elemen baru
- **remove()**, untuk menghapus nilai pada ArrayList

UNIVERSITAS ISLAM INDONESIA

Operasi ArrayList

Pembuatan Class dan Atributnya

- Pembuatan class ArrayList beserta atributnya:
  - **<E>**: *Generic Types*, secara umum, ini memungkinkan *types* dalam Java menjadi parameter saat mendefinisikan sebuah kelas (*classes*), antarmuka (*interfaces*), dan metode (*methods*). ***Misalnya*** membuat ArrayList **String** atau **Integer**
  - **arrayList**: array bertipe objek untuk menyimpan informasi
  - **DEFAULT_CAPACITY**: **kapasitas** default dari ArrayList
  - **size**: variabel untuk menyimpan informasi jumlah elemen (**ukuran**) di ArrayList

```
public class ArrayList<E> {
    private Object[] arrayList;
    private static final int DEFAULT_CAPACITY = 10;
    private int size;
```

Pembuatan Constructor

```
public ArrayList() {
   this(DEFAULT_CAPACITY);
   this.size = 0;
}

public ArrayList(int capacity) {
   if (capacity <= 0) {
      System.out.println("The capacity must be greater than 0.");
      return;
   }
   this.arrayList = new Object[capacity];
   this.size = 0;
}
```

Operasi `size()` dan `isEmpty()`

- **size()** - Operasi yang digunakan untuk mencari ukuran ArrayList

```
public int size() {
    return this.size;
}
```

- **isEmpty()** - Operasi yang digunakan untuk memeriksa apakah ArrayList kosong atau tidak

```
public boolean isEmpty() {
    return this.size == 0;
}
```

Operasi `indexOf(Object o)`

- Mengembalikan index kemunculan pertama dari elemen yang dicari, atau mengembalikan nilai -1 jika elemen yang dicari tidak ada dalam ArrayList.

```
public int indexOf (Object obj) {
   for (int i = 0; i < this.size(); i++) {
      if (obj.equals(this.arrayList[i])) {
         return i;
      }
   }
   return -1;
}
```

Operasi `contains(Object o)`

- Operasi yang digunakan untuk memeriksa apakah suatu nilai ada dalam ArrayList atau tidak.
- Operasi ini akan mengembalikan nilai benar (**true**) jika ArrayList berisi elemen yang dicari.

```
public boolean contains(Object obj) {
   return indexOf(obj) >= 0;
}
```

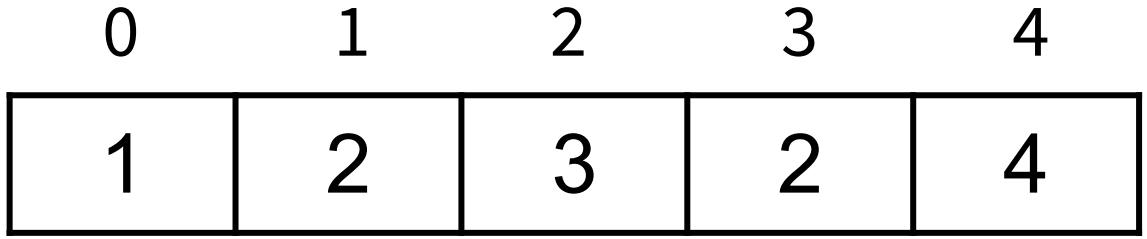
Operasi `isFull()`

- Operasi yang digunakan untuk memeriksa apakah kapasitas ArrayList sudah penuh atau belum

```
private boolean isFull() {
   return this.arrayList.length == this.size;
}
```

Operasi `get(int index)`

- Operasi yang digunakan untuk mengambil elemen pada index tertentu pada ArrayList

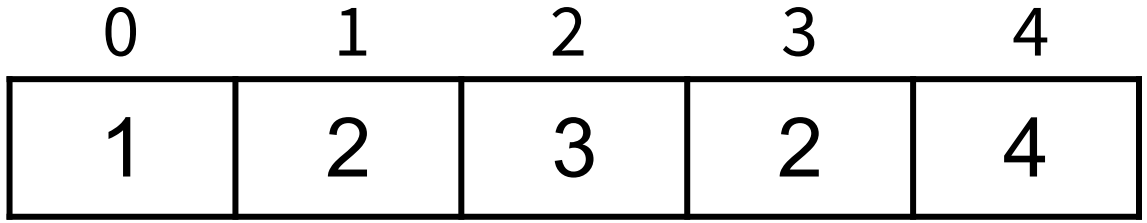



```
public Object get(int index) {
        Object element = null;

        if (index < 0 || index >= this.size()) {
            System.out.println("Index out of bounds");
            System.exit(-1);
        }else{
            element = this.arrayList[index];
        }
        return element;
    }
```

Operasi `set(int index, E obj)`

- Operasi yang digunakan untuk mengganti nilai pada index tertentu.
- Jika index melebihi index terbesar pada ArrayList, akan memunculkan error ***IndexOutOfBoundsException***

```
public void set(int index, Object obj){
        if (index < 0 || index >= this.size()) {
            System.out.println("Index out of bounds");
            System.exit(-1);
        }else{
            this.arrayList[index] = obj;
        }
    }
```



Operasi `clear()`

- Operasi yang digunakan untuk menghapus semua elemen pada ArrayList.

```
public void clear() {
   if(this.size() > 0){
      this.arrayList = null;
      this.size = 0;
      this.arrayList = new Object[ArrayList.DEFAULT_CAPACITY];
   }
}
```

Operasi `resizeArray()`

1. Operasi yang digunakan untuk menambahkan kapasitas ArrayList
2. Operasi ini akan dieksekusi setiap kali ada proses **penambahan elemen baru** dan kondisi kapasitas ArrayList **sudah penuh**
3. Pada operasi ini akan dibuat ***ArrayList sementara*** yang kapasitasnya

   `kapasitas_arraylist_asli + (kapasitas_arraylist_asli >> 1)`

   misalnya → `5 + (5 >> 1)`
4. Kemudian semua elemen pada **ArrayList asli** akan dipindahkan ke ***ArrayList sementara***
5. **ArrayList asli** akan diberi nilai **null** dan kapasitasnya **diganti** dengan kapasitas baru yang sama dengan ukuran ***ArrayList sementara***
6. Selanjutnya semua elemen ***ArrayList sementara*** akan dimasukkan ke **ArrayList asli**

Operasi resizeArray()

```
private void resizeArray() {
    int oldCapacity = this.arrayList.length;
    //kapasitas baru
    int newCapacity = oldCapacity + (oldCapacity >> 1);
    //pembuatan ArrayList sementara dengan kapasitas baru
    Object[] tempArray = new Object[newCapacity];
    //pemindahan elemen ArrayList asli ke ArrayList sementara
    for (int i = 0; i < this.size(); i++) {
        tempArray[i] = this.arrayList[i];
    }
    this.arrayList = null;
    this.arrayList = new Object[tempArray.length];
    this.arrayList = tempArray;
}
```

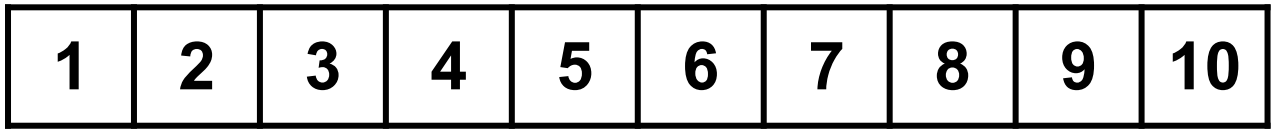
Operasi `resizeArray()`

oldCapacity = 10
newCapacity = oldCapacity + (oldCapacity >> 1)
newCapacity = 10+5
newCapacity = 15

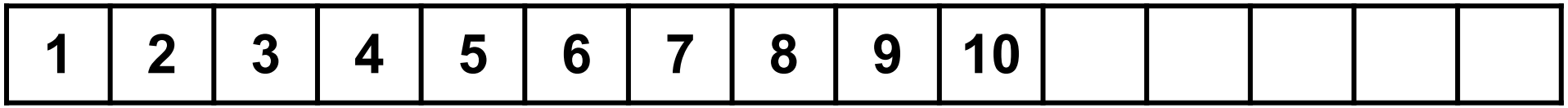
oldCapacity = 10 = 1010

1010 >> 1 = 101 = **5**

old ArrayList



new ArrayList



Operasi `add(E element)`

Ada 2 macam operasi **add()** pada ArrayList:

- **add(E element)**
  - Menambahkan elemen baru **di akhir** ArrayList
  - Sebelum ditambahkan, akan dicek apakah kapasitas ArrayList sudah penuh atau belum.
  - Jika sudah **penuh** maka **kapasitas** ArrayList akan **ditambah**
  - Kemudian **elemen baru** akan **ditambahkan** di **index terakhir + 1** dan ukuran ArrayList akan ditambahkan 1

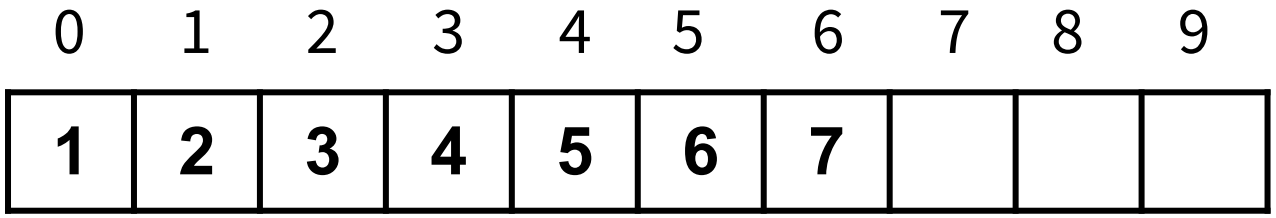
Operasi add(E element)

```
public void add(E obj) {
   //pengecekan kapasitas ArrayList sudah penuh atau belum
   if (this.isFull()) {
      //jika sudah penuh maka kapasitasnya akan ditambah
      this.resizeArray();
   }
   //penambahan elemen baru di index terakhir
   this.arrayList[this.size] = obj;
   //ukuran ArrayList ditambah 1
   this.size++;
}
```

Operasi add ( )

add(11)

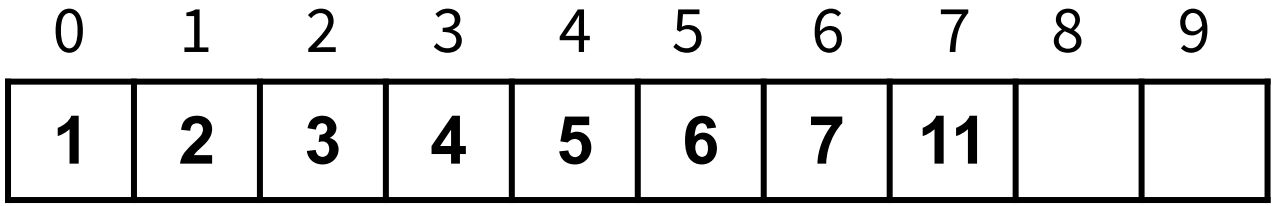
size = 7, index terakhir = 6



full == **false**

masukkan 11 ke index [7]

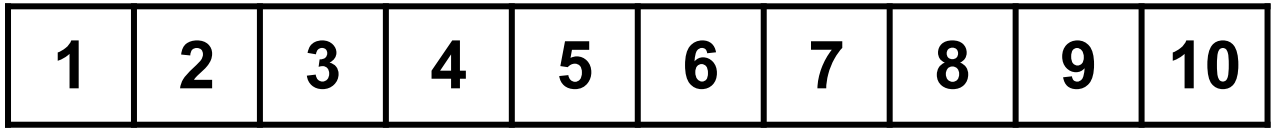
```
this.arrayList[this.size] = obj;
```



Operasi add ( )

add(11)

size = 10, index terakhir = 9



full == **true**

resize ArrayList

masukkan 11 ke index [10]

```
this.arrayList[this.size] = obj;
```

Operasi `add(int index, E element)`

- **add(int index, E element)**
  - Menambah/menyisipkan elemen baru pada posisi/index tertentu
  - **Index terbesar** yang dapat disisipi elemen baru adalah index yang **besarnya sama** dengan **ukuran ArrayList saat ini**.
    - Jika penambahan dilakukan di **index terbesar**, akan dicek apakah kapasitas ArrayList sudah penuh atau belum.
    - ***Jika sudah penuh*** maka **kapasitas** ArrayList akan **ditambah**
    - Kemudian elemen baru ditambahkan di index terbesar
  - Jika index akan disisipi **lebih besar** dari ukuran ArrayList maka akan terjadi error ***IndexOutOfBoundsException***.

Operasi `add(int index, E element)`

- **add(int index, E element)**
  - Jika penambahan dilakukan di index yang besarnya kurang dari ukuran ArrayList saat ini maka:
    - Elemen yang ada *di index yang akan disisipi disimpan di variabel sementara*
    - Elemen yang **baru** akan ditambahkan ke index tersebut
    - Elemen yang lama (yang ada di variabel sementara) akan diletakkan di **indexnya + 1**
    - Kemudian akan ada ***proses penataan ulang*** urutan elemen-elemen lainnya yang index-nya lebih dari indek yang sudah disisipi.
  - Jika penambahan selesai maka ukuran ArrayList akan ditambah 1

Operasi add(int index, E element)

```
public void add(int index, E obj) {
   if (index < 0 || index > this.size()) {
      System.out.println("Index out of bounds");
      System.exit(-1);
   }else{
      //pengecekan kapasitas ArrayList
      if (this.isFull()) {
         this.resizeArray();
      }

      if(index == this.size()){
         //penambahan di akhir ArrayList
         this.arrayList[index] = obj;
      }else{
```

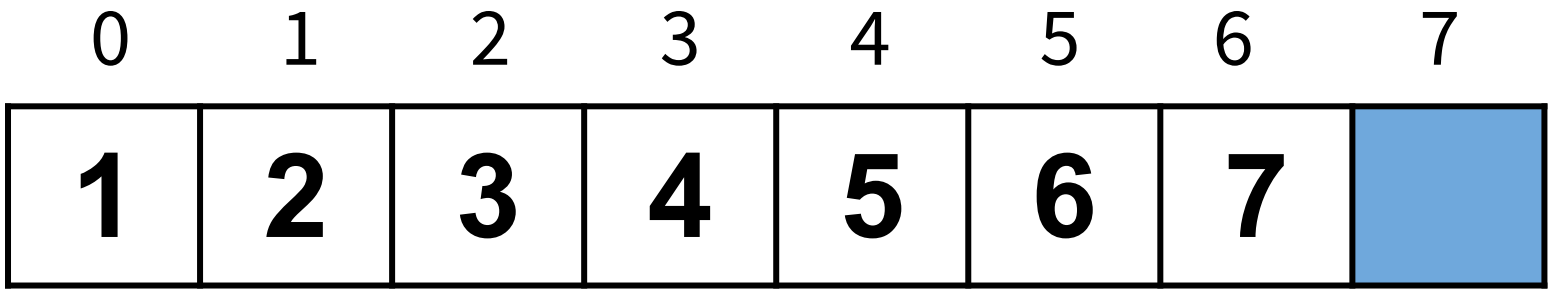
Operasi add(int index, E element)

```
            Object temp = this.arrayList[index];
            //Penyisipan elemen baru ke index
            this.arrayList[index] = obj;

            Object temp2;
            //proses pengurutan ulang index elemen
            for (int i = index; i < this.size(); i++) {
                temp2 = this.arrayList[i + 1];
                this.arrayList[i + 1] = temp;
                temp = temp2;
            }
        }
        //penambahan ukuran ArrayList
        this.size++;
    }
}
```

Operasi `add(int index, E element)`

index = 7
obj = 10 (elemen baru)
size = 7

add(7, 10)

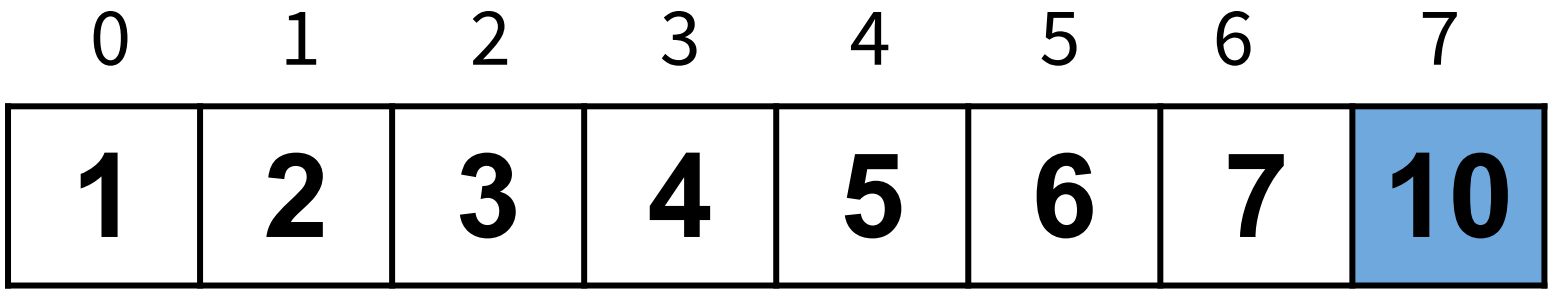



```
if(index == this.size()){
   //penambahan di akhir ArrayList
   this.arrayList[index] = obj;
}
```

Operasi `add(int index, E element)`

index = 7
obj = 10 (elemen baru)
size = 7

add(7, 10)



size = 8

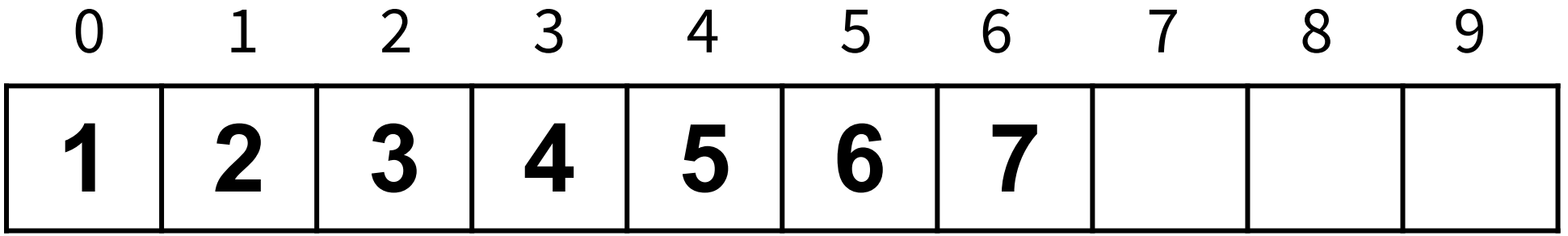
```
if(index == this.size()){
   //penambahan di akhir ArrayList
   this.arrayList[index] = obj;
}
```

```
this.size++;
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

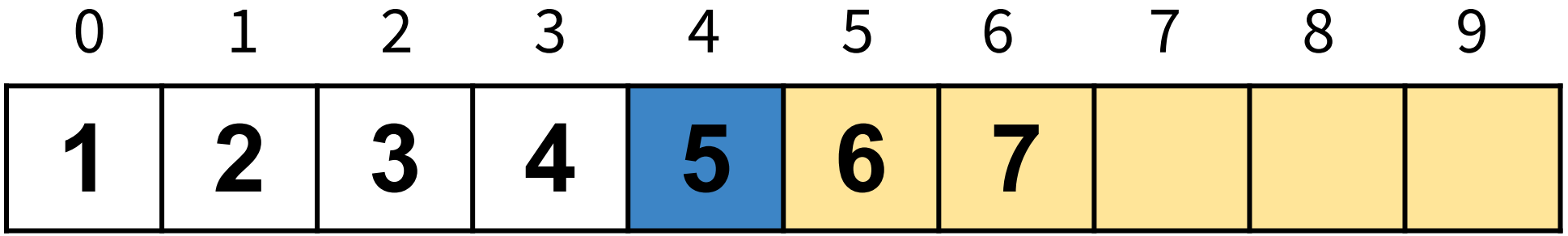
add(4, 10)



Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



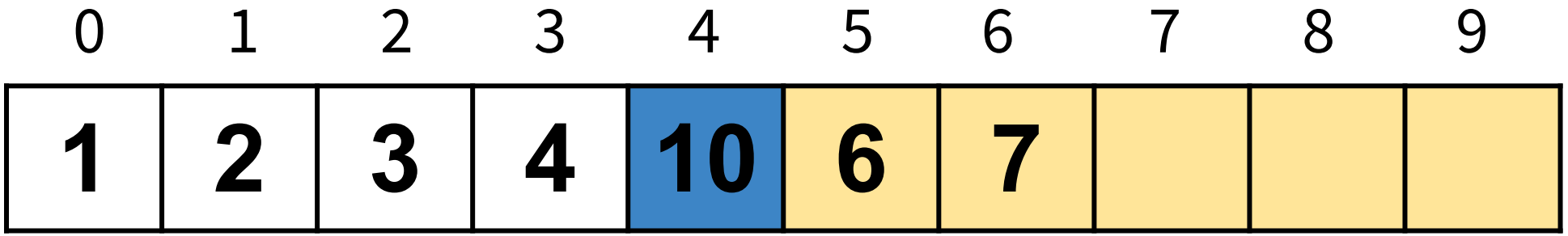
temp **5**

```
Object temp = this.arrayList[index];
this.arrayList[index] = obj;
Object temp2;
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



temp 5

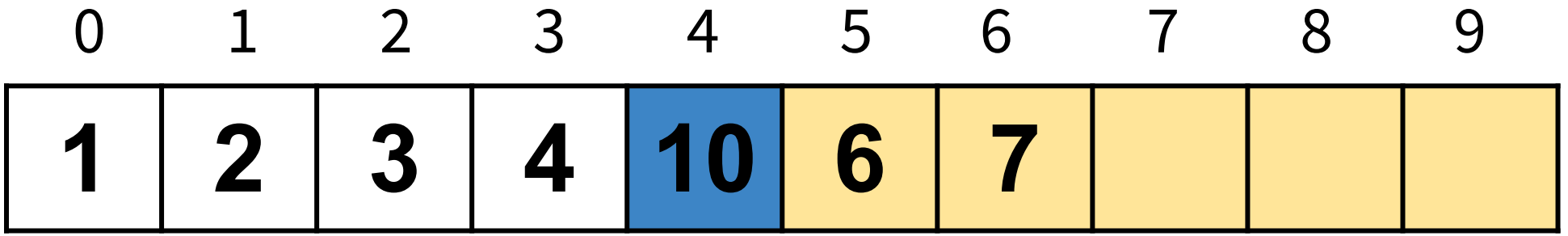
temp2

```
Object temp = this.arrayList[index];
this.arrayList[index] = obj;
Object temp2;
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan dimulai dari **i = 4** s/d **i = 6** (3 kali perulangan)

temp 5

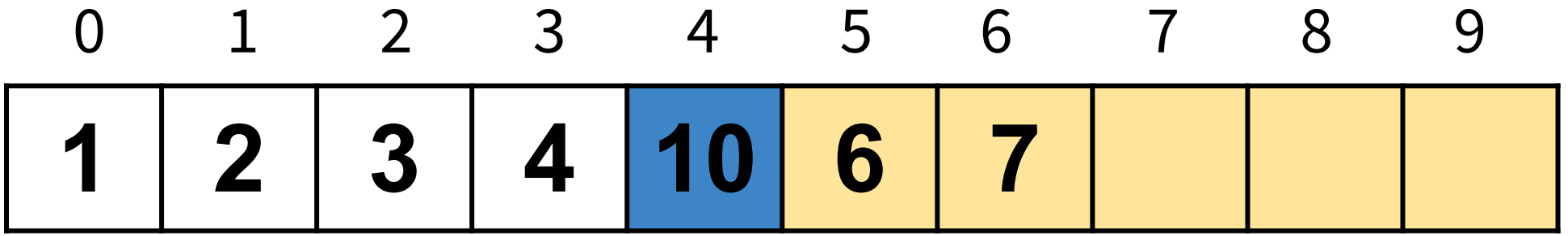
temp2

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-1**

i = 4

temp **5**

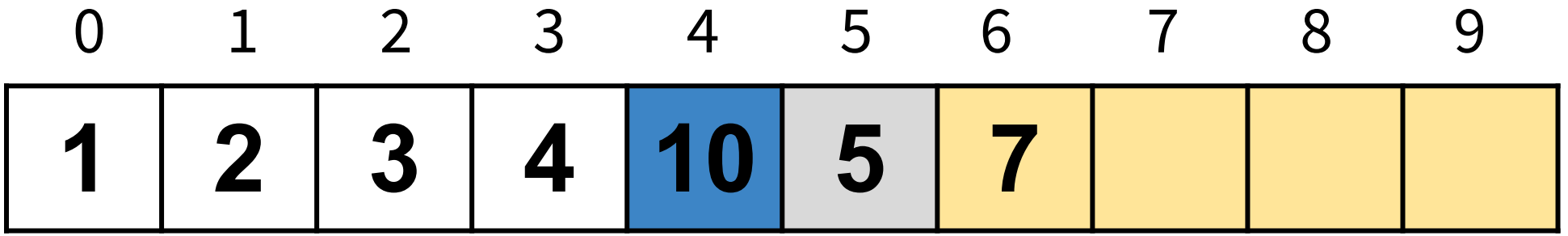
temp2 **6**

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-1**

i = 4

temp

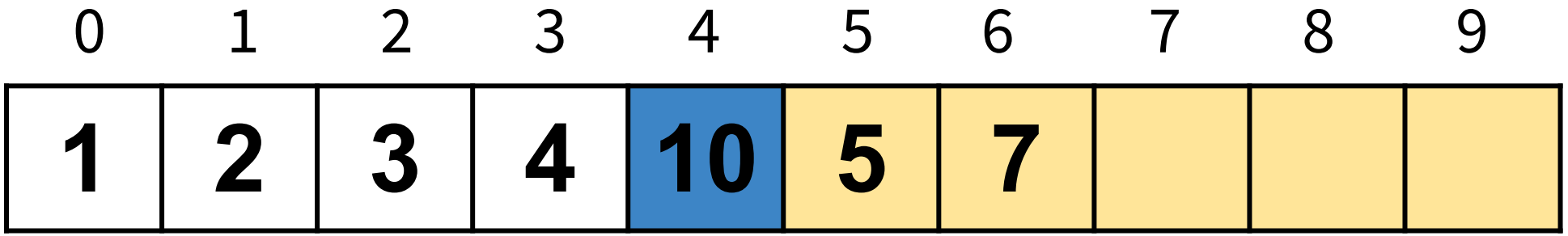
temp2 **6**

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-1**

i = 4

temp **6**

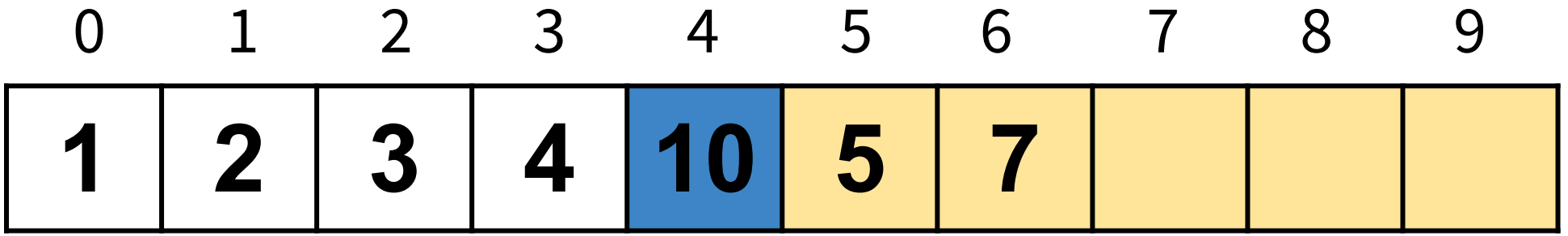
temp2

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-2**

i = 5

temp **6**

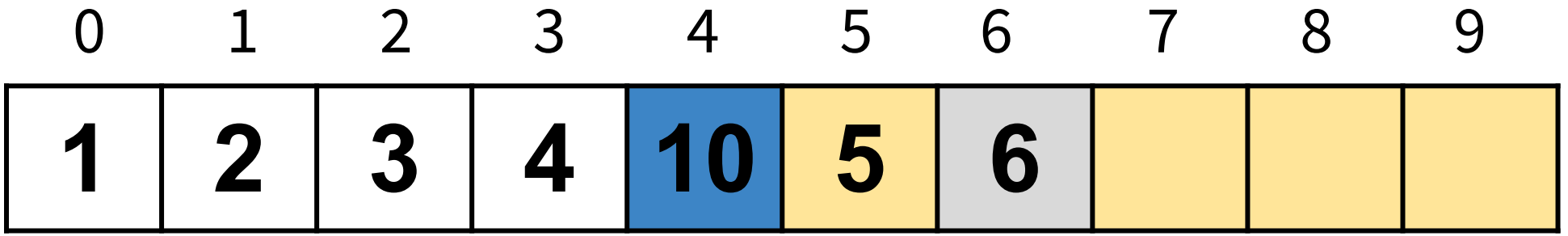
temp2 **7**

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-2**

i = 5

temp

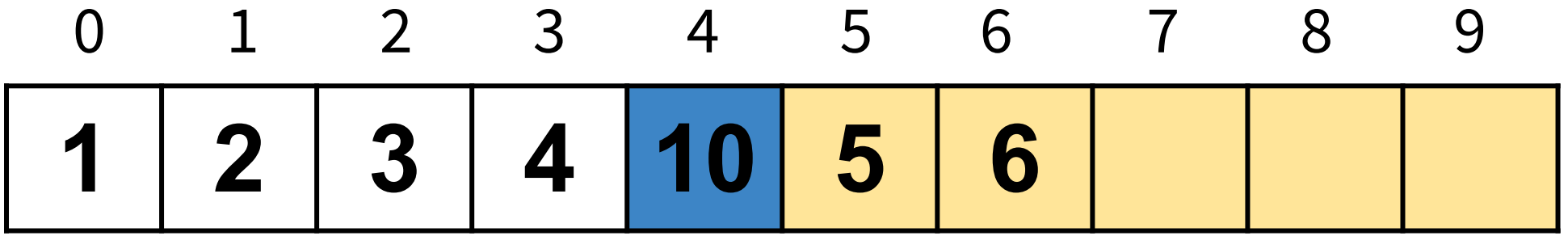
temp2 **7**

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

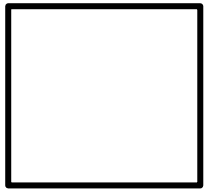
add(4, 10)



perulangan **ke-2**

i = 5

temp 

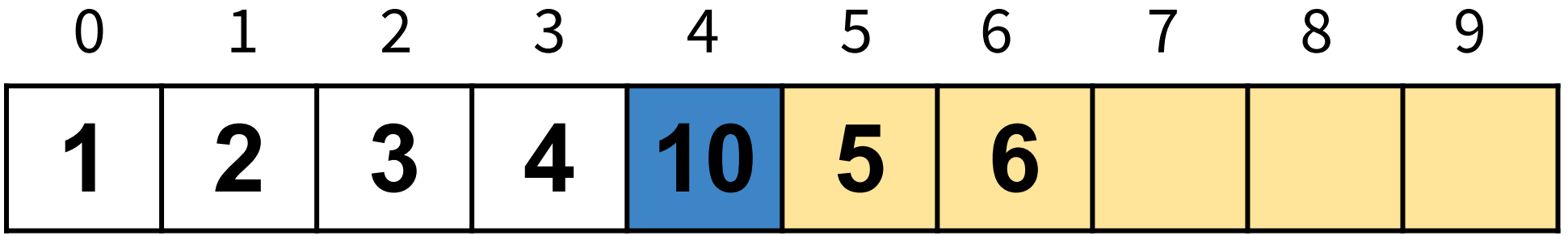
temp2 

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-3**

i = 6

(terakhir)

temp | **7**

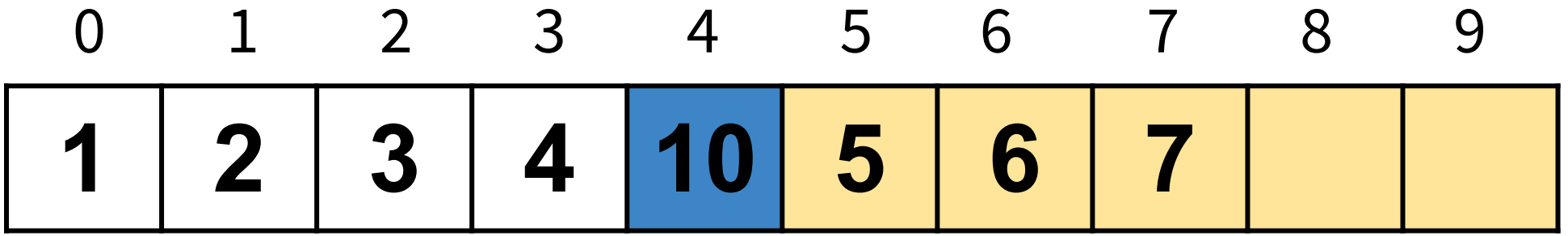
temp2 | **null**

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-3**

i = 6

(terakhir)

temp

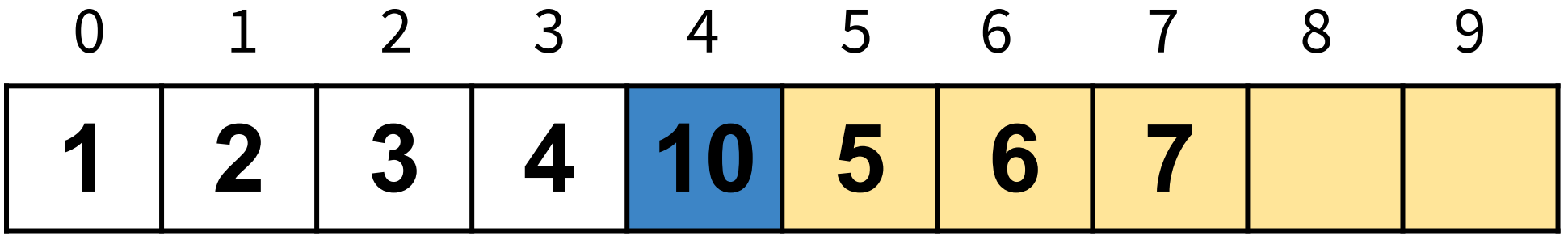
temp2 **null**

```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

index = 4
obj = 10 (elemen baru)
size = 7

add(4, 10)



perulangan **ke-3**

i = 6

(terakhir)

temp null

temp2

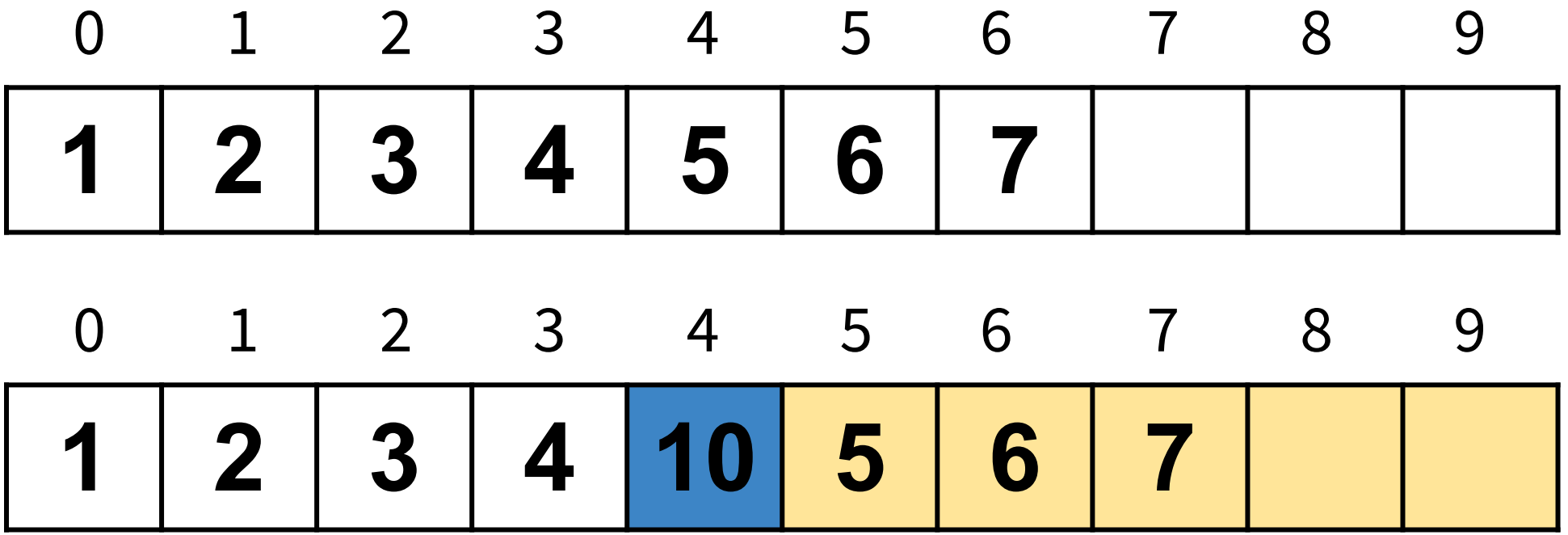
```
for (int i = index; i < this.size(); i++)
{
    temp2 = this.arrayList[i + 1];
    this.arrayList[i + 1] = temp;
    temp = temp2;
}
```

Operasi `add(int index, E element)`

`this.size++`

(size baru = 8)

add(4, 10)



Operasi `shiftArray(int index)`

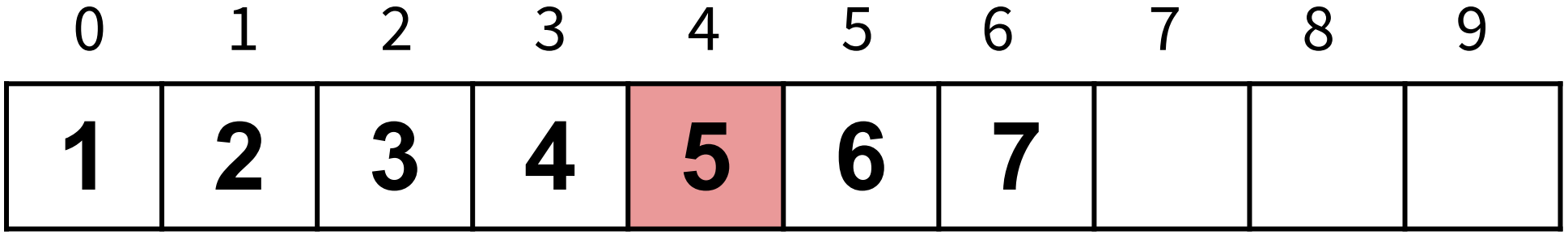
- Operasi ini digunakan untuk mengatur ulang ArrayList ketika ada proses ***penghapusan sebuah elemen***.
- Dimulai dengan membuat ***ArrayList sementara*** yang kapasitasnya sama dengan ***ArrayList asli*** dan digunakan ntuk menyimpan hasil pengaturan ulang ArrayList.
- Kemudian semua elemen ***ArrayList asli*** akan dipindahkan ke ***ArrayList sementara*** kecuali elemen yang akan dihapus.
- Setelah selesai ukuran ArrayList asli akan dikurangi 1
- Berikutnya **ArrayList asli** diberi nilai **null** kemudian semua elemen di ***ArrayList sementara*** dipindahkan ke **ArrayList asli**

Operasi shiftArray(int index)

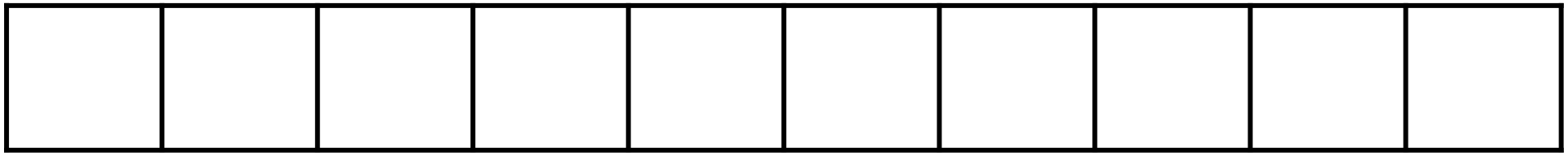
```
private void shiftArray(int index) {
        Object[] tempArray = new Object[this.arrayList.length];
        int indexTemp = 0;

        for (int j = 0; j < this.size(); j++) {
            if(index != j){
                tempArray[indexTemp] = this.arrayList[j];
                indexTemp++;
            }
        }
        this.size--;
        this.arrayList = null;
        this.arrayList = tempArray;
    }
```

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4




```
Object[] tempArray = new Object[this.arrayList.length];
int indexTemp = 0;
```

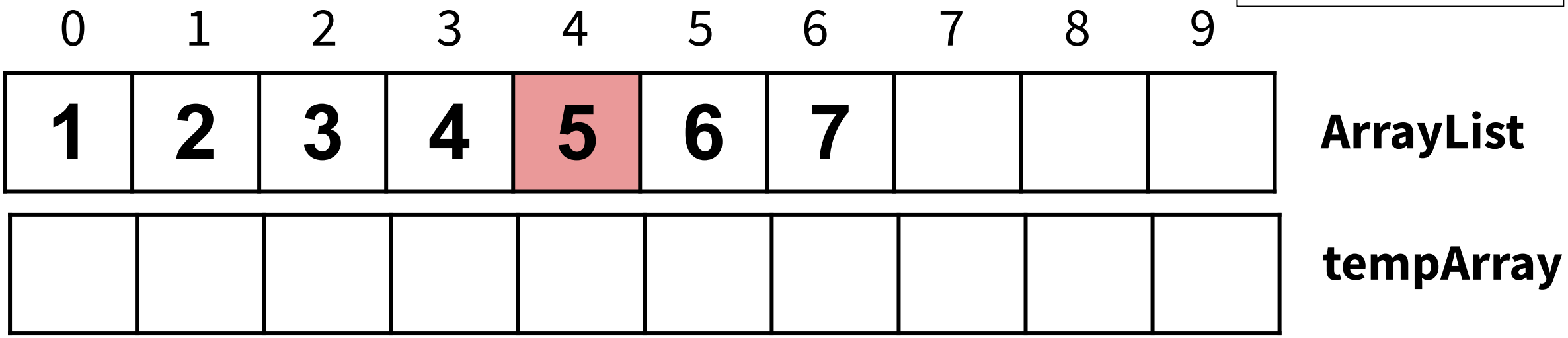


indexTemp = 0

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

Perulangan **ke-1**

j = 0

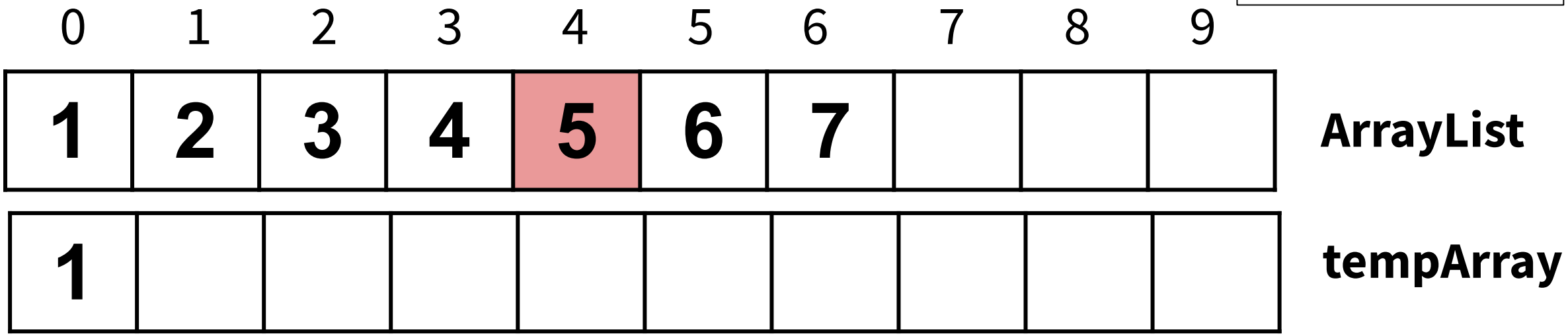
indexTemp = 0

`(4 != 0)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

indexTemp berikutnya = 1

Perulangan **ke-1**

j = 0

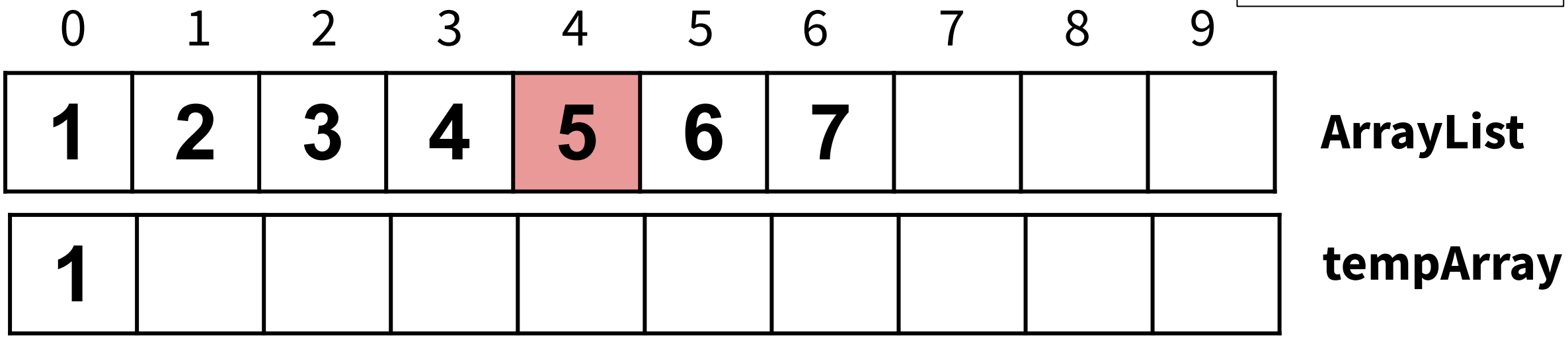
indexTemp = 0

`(4 != 0)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

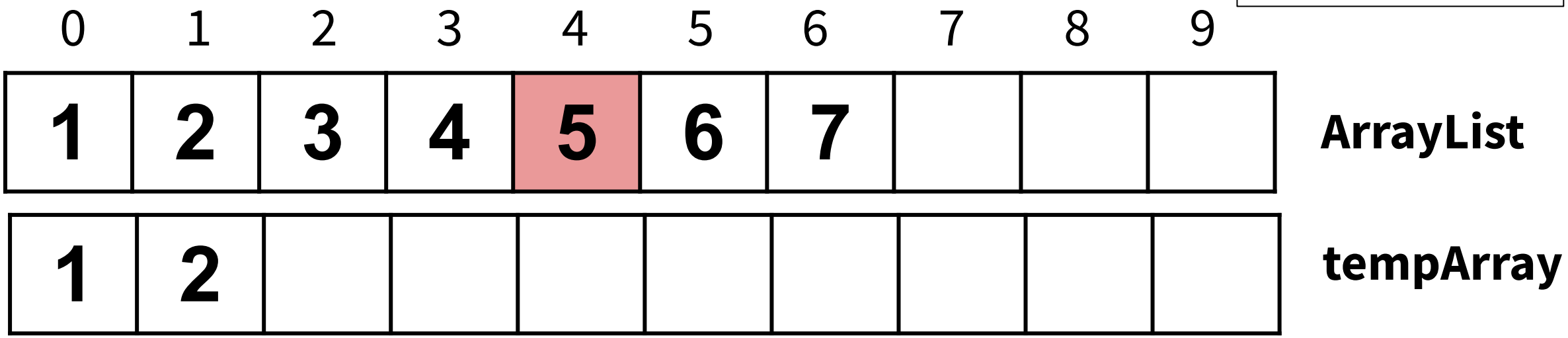
Perulangan **ke-2**
j = 1
indexTemp = 1

`(4 != 1)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

indexTemp berikutnya = 2

Perulangan **ke-2**

j = 1

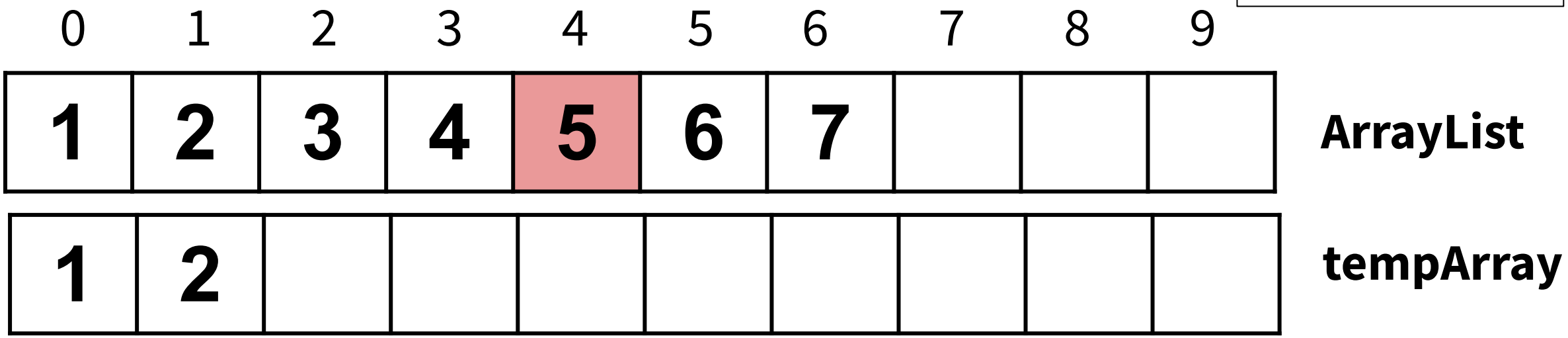
indexTemp = 1

`(4 != 1)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

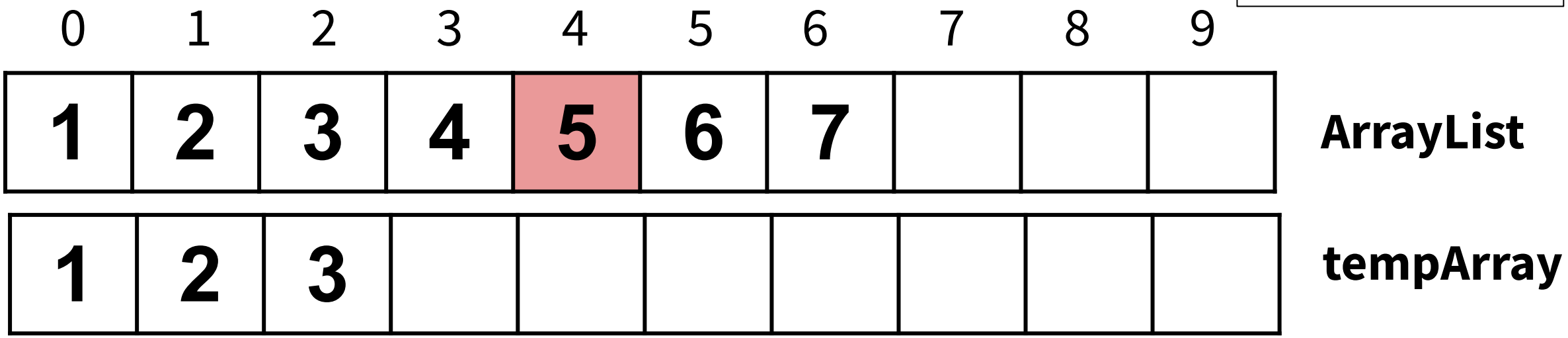
Perulangan **ke-3**
j = 2
indexTemp = 2

`(4 != 2)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

indexTemp berikutnya = 3

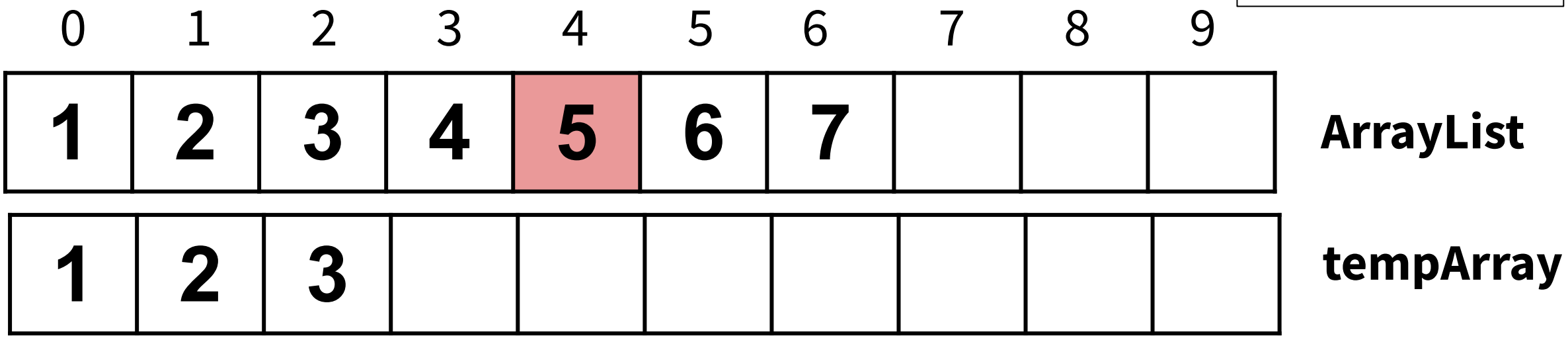
Perulangan **ke-3**
j = 2
indexTemp = 2

`(4 != 2)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

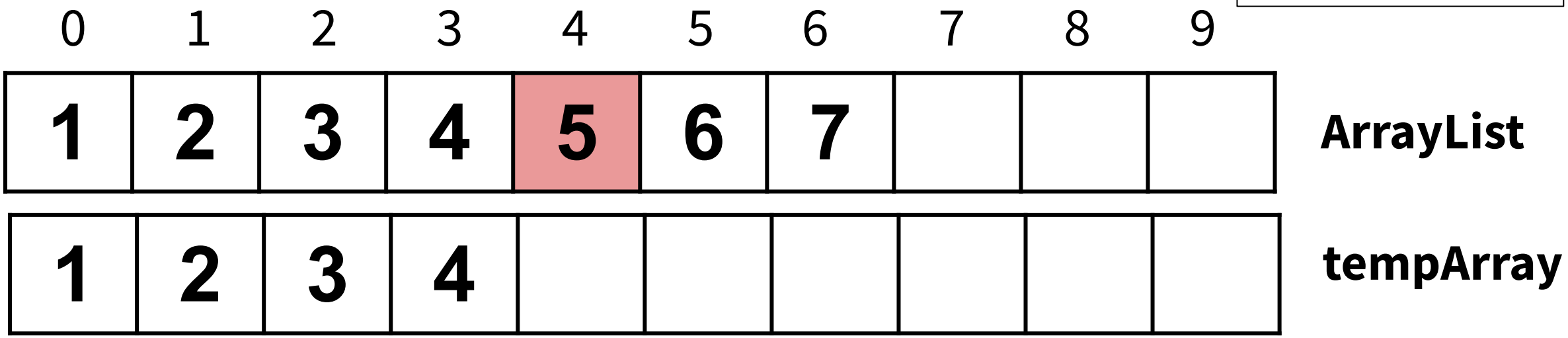
Perulangan **ke-4**
j = 3
indexTemp = 3

`(4 != 3)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

indexTemp berikutnya = 4

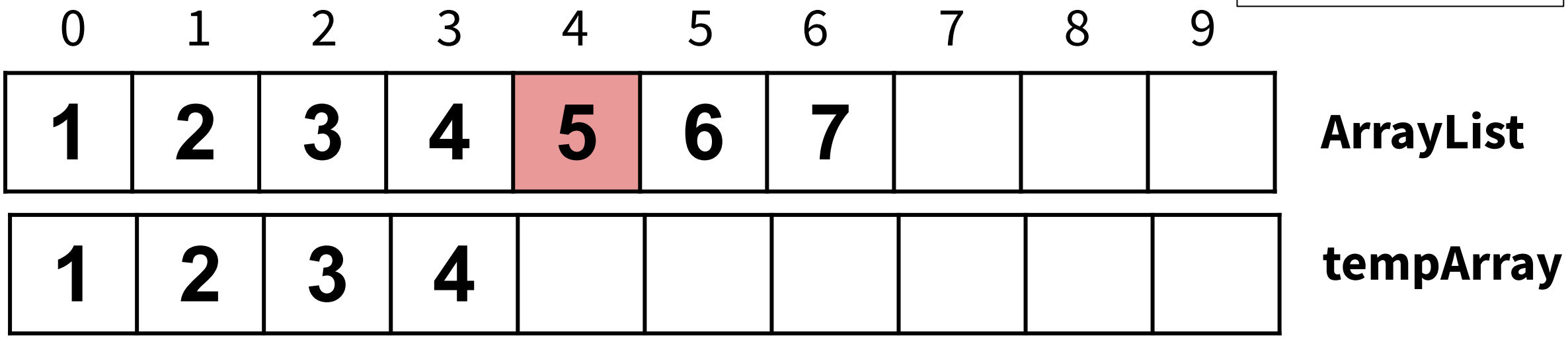
Perulangan **ke-4**
j = 3
indexTemp = 3

`(4 != 3)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

indexTemp berikutnya = tetap

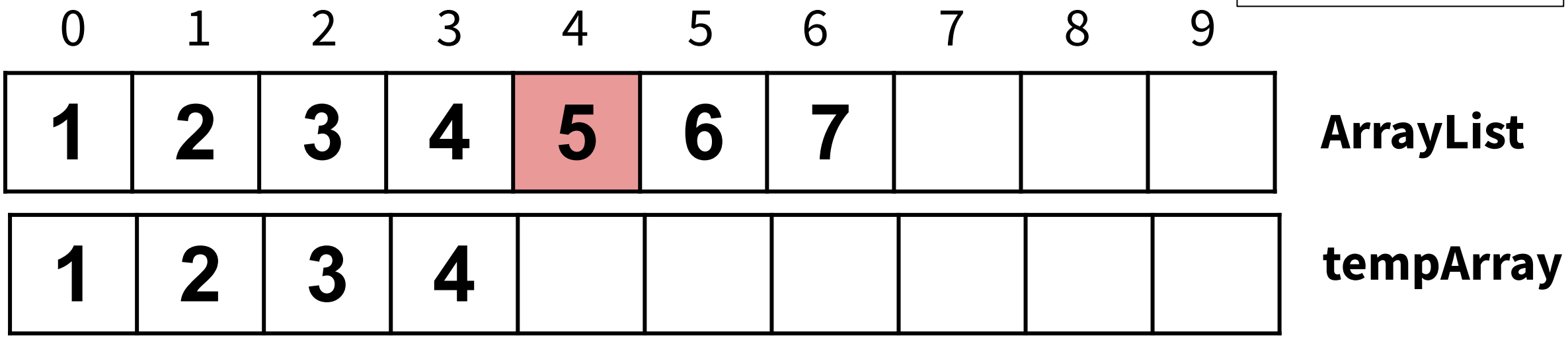
Perulangan **ke-5**
j = 4
indexTemp = 4

`(4 != 4)` → **false**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

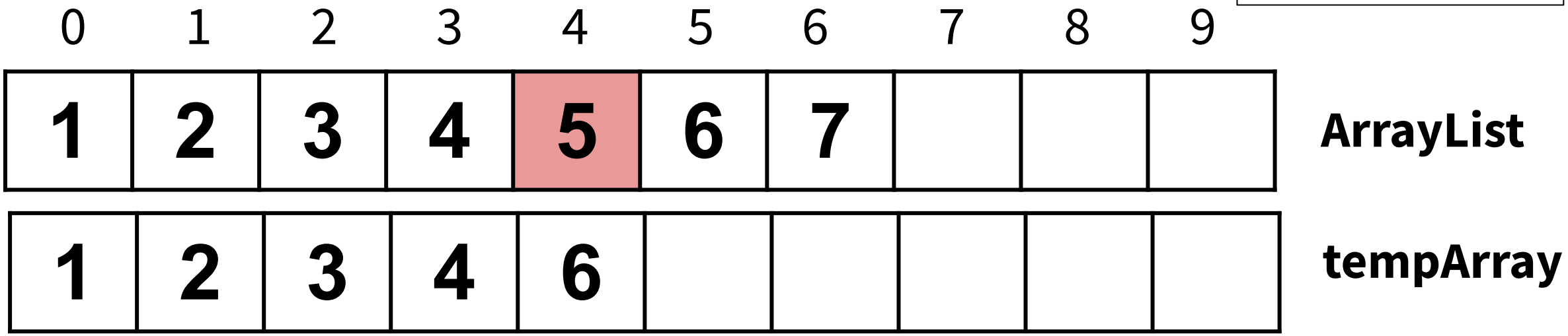
Perulangan **ke-6**
j =5
indexTemp = 4

`(4 != 5)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

indexTemp berikutnya = 5

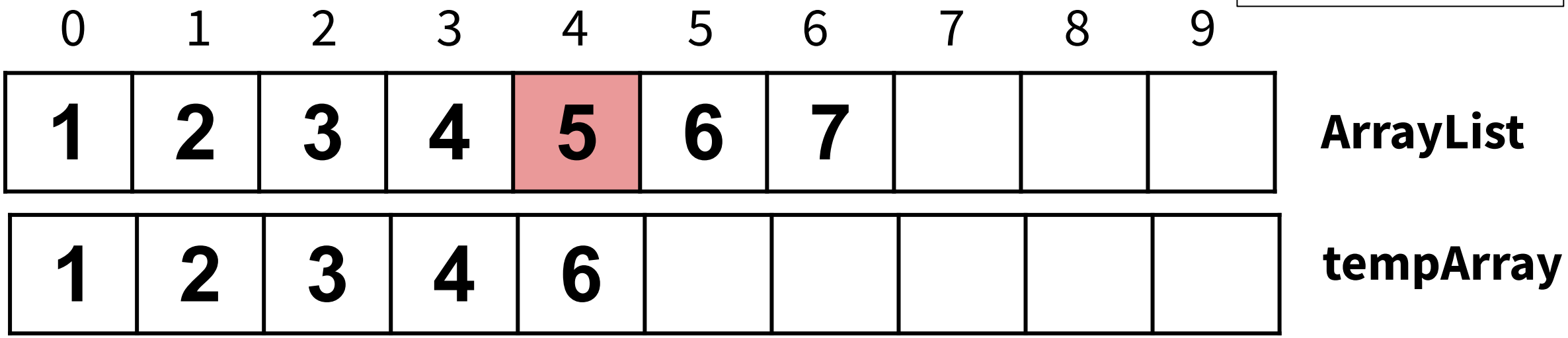
Perulangan **ke-6**
j =5
indexTemp = 4

`(4 != 5)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

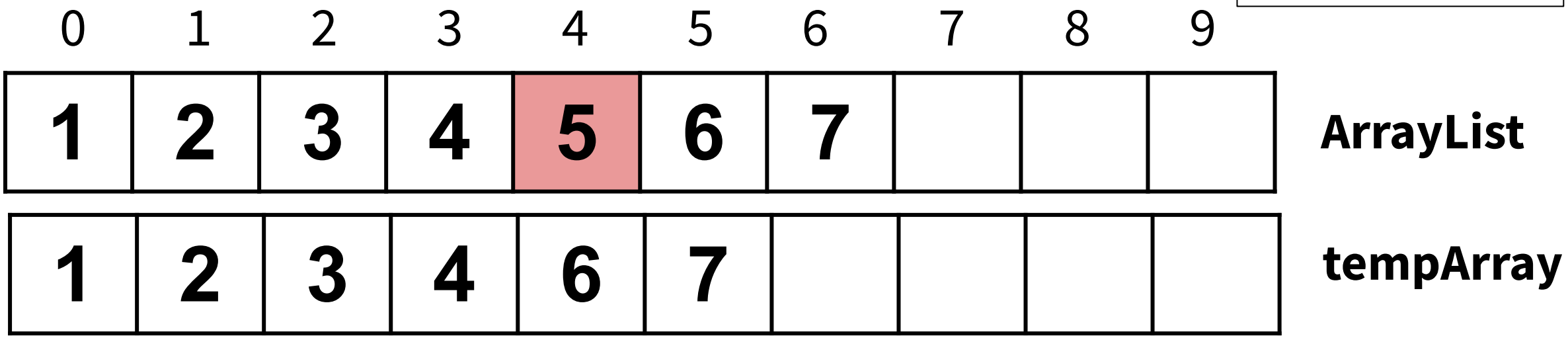
Perulangan **ke-7**
j =6
indexTemp = 5

`(4 != 6)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
for (int j = 0; j < this.size(); j++) {
    if(index != j){
        tempArray[indexTemp] = this.arrayList[j];
        indexTemp++;
    }
}
```

indexTemp berikutnya = 6

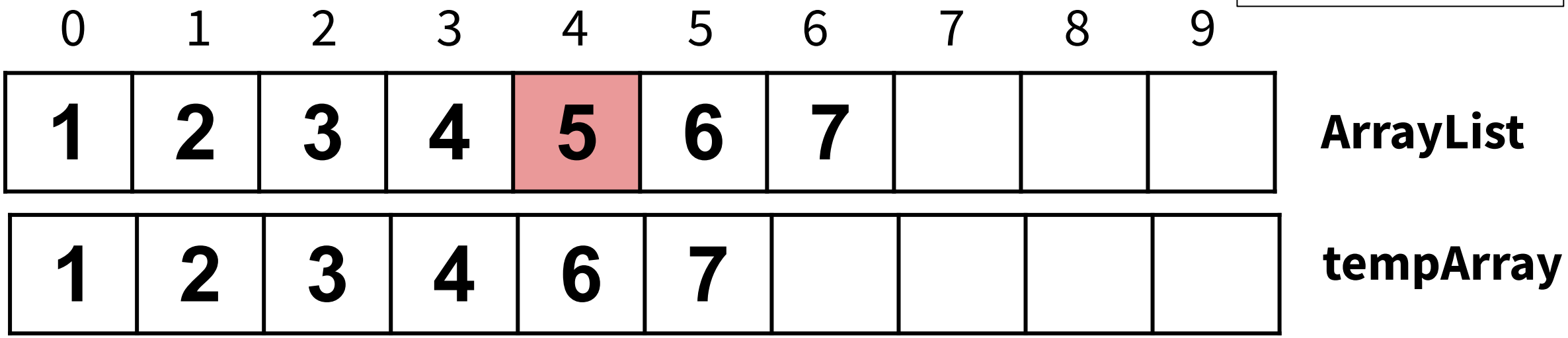
Perulangan **ke-7**
j =6
indexTemp = 5

`(4 != 6)` → **true**

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```



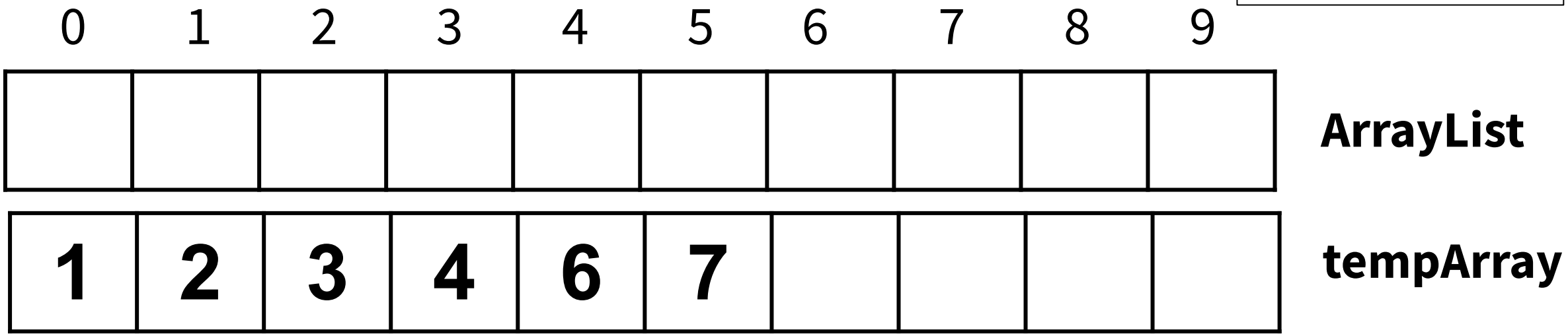

```
this.size--;
this.arrayList = null;
this.arrayList = tempArray;
```

size baru = 6

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```



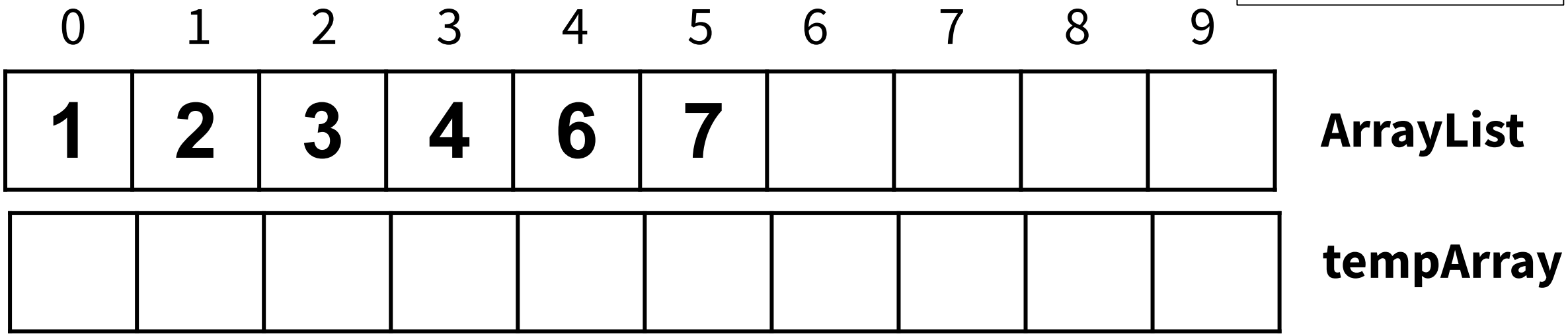

```
this.size--;
this.arrayList = null;
this.arrayList = tempArray;
```

size baru = 6

Operasi `shiftArray(int index)`

remove(4) → menghapus elemen indeks ke-4

```
index = 4
size = 7
```




```
this.size--;
this.arrayList = null;
this.arrayList = tempArray;
```

size baru = 6

Operasi `remove(Object o)`

Ada 2 macam operasi **remove()** pada ArrayList:

- **remove(Object o)**
  - Menghapus berdasarkan objeknya
  - Menghapus kemunculan pertama dari elemen yang ditentukan dari ArrayList, jika ada.
  - Proses dimulai dengan mencari index elemen yang akan dihapus berdasarkan nilainya
  - Jika sudah ketemu maka index tersebut akan dijadikan kunci untuk proses pengapusan

```
public void remove (Object obj) {
   int indexFound = this.indexOf(obj);
   if(indexFound != -1){
      this.shiftArray(indexFound);
   }
}
```

Operasi `remove(int index)`

- **remove(int index)**
  - Menghapus berdasarkan index dari elemen yang akan dihapus
  - Jika index elemen yang akan dihapus lebih dari index elemen terakhir ArrayList maka akan muncul error ***IndexOutOfBoundsException***
  - Index tersebut akan dijadikan kunci untuk proses pengapusan

```
public void remove (int index) {
   if (index < 0 || index >= this.size()) {
      System.out.println("Index out of bounds");
      System.exit(-1);
   }else{
      this.shiftArray(index);
   }
}
```