BAB 6
PEMANFAATAN ARRAY SEBAGAI TUMPUKAN (STACK) DAN ANTRIAN (QUEUE)

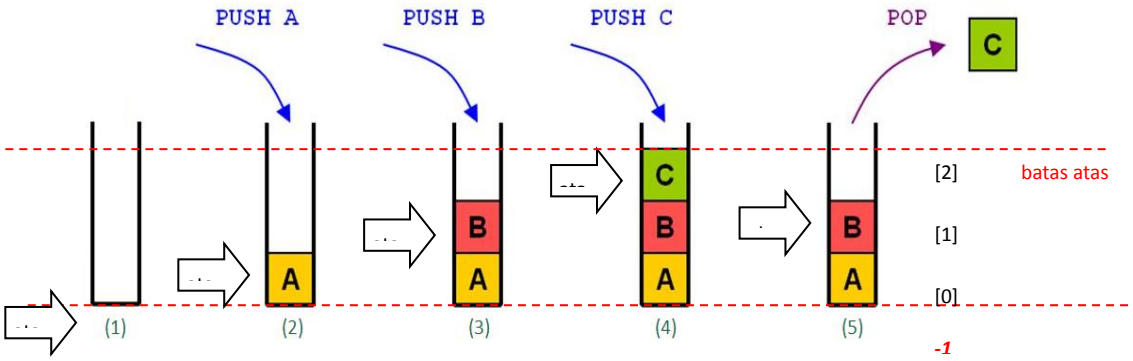
Tujuan

Mahasiswa dapat mengimplementasikan tumpukan dan antrian untuk berbagai keperluan dengan menggunakan bahasa pemrograman Java.

6.1. Stack (Tumpukan)

Tumpukan atau stack adalah teknik peletakan data dimana seolah-olah data yang satu diletakkan 'diatas' data yang lain. Dalam tumpukan semua penyisipan dan penghapusan data dilakukan hanya melalui satu pintu yang disebut top (puncak) tumpukan. Tumpukan dapat dibayangkan seperti tumpukan buku dimana hanya buku teratas yang dapat diperoleh kembali dengan satu langkah. Buku-buku yang terdapat dibawah hanya dapat diambil setelah buku yang berada diatasnya diambil (dikeluarkan) terlebih dahulu. Dengan bentuk yang demikian tumpukan dapat dikatakan sebagai struktur data yang bersifat **LIFO (Last In First Out**).

Dalam tumpukan terdapat 2 buah operasi data yaitu (1) operasi **push** dan (2) operasi **pop**. Operasi **push** adalah operasi untuk menambahkan sebuah data ke dalam tumpukan, sedangkan operasi **pop** adalah operasi untuk mengambil (menghapus) data dari dalam tumpukan.



Gambar 7.1 Ilustrasi Tumpukan

Dalam implementasinya tumpukan selalu memiliki kapasitas yaitu batas maksimum di mana tumpukan mampu menampung data.

Ada 2 kondisi yang menjadi batasan sebuah tumpukan yaitu :

1. Kondisi **Tumpukan penuh**.

Kondisi tumpukan penuh adalah kondisi di mana nilai `atas` = nilai `batasAtas`. Apabila kondisi ini terjadi maka tumpukan tidak lagi dapat menerima pemasukan data baru (push data baru)

2. Kondisi **Tumpukan kosong**.
   Kondisi tumpukan kosong adalah kondisi di mana nilai `atas` = -1. Apabila kondisi ini terjadi maka kita tidak lagi dapat melakukan pengambilan/penghapusan data (pop data) dari tumpukan

Program tumpukan dapat dilihat pada program 7.1 berikut ini.

```
public class program_tumpukan
{   public static int N = 5;
    public static int atas = -1;

    public static void PUSH (String tumpukan[], String data)
    {   if (atas == N-1) //jika tumpukan penuh
        {   System.out.println("maap, tumpukan penuh, PUSH " + data
                    + "  tidak dapat dilakukan");
        }
        else //jika tumpukan tidak penuh
        {   atas = atas + 1;
            tumpukan[atas] = data;
            System.out.println("PUSH " + data + " berhasil..");
        }
    }
    public static String POP (String tumpukan[])
    {   String hasil;
        if (atas < 0 ) //jika tumpukan kosong
        {   hasil = "TUMPUKAN KOSONG, POP GAGAL DILAKUKAN";
        }
        else //jika tumpukan tidak kosong
        {   hasil = tumpukan[atas];
            atas = atas - 1;
        }
        return (hasil);
    }
    public static void lihatTumpukan(String tumpukan[])
    {   System.out.println("");
        System.out.println("--TUMPUKAN:-");
        for (int i=atas;  i>=0;  i--)
        {   System.out.println(tumpukan[i]);
        }
        System.out.println("--akhir tumpukan-");
        System.out.println("");
    }

    public static void main (String[] args)
    {   String tumpukan[] = new String[10];
        ............;
        ............;
        ............;
        ............;
    }
}
```

Program 7.1

6.2. Queue (Antrian)

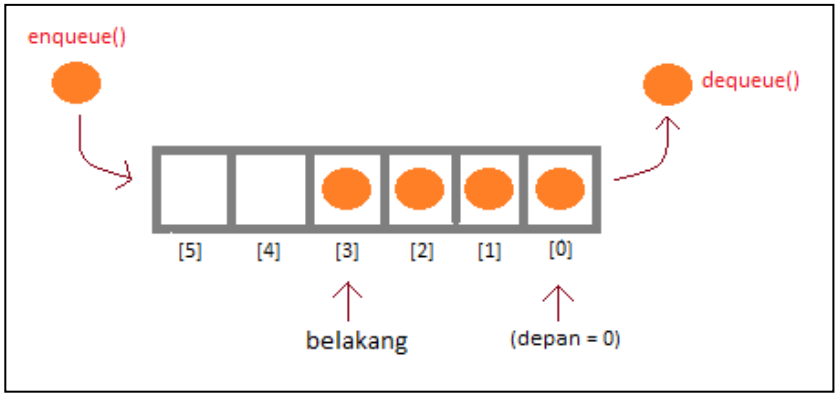
Antrian atau Queue adalah teknik peletakan data dimana seolah-olah data yang satu diletakkan ‘dibelakang’ data yang lain. Dalam antrian semua penyisipan dan penghapusan data dilakukan melalui dua pintu yaitu belakang dan depan. Pintu belakang digunakan sebagai jalan masuk data, sedangkan pintu depan digunakan sebagai jalan keluar data.

Antrian dapat dibayangkan seperti antrian para perima zakat. Orang yang ingin mendapatkan zakat harus mengantri dengan cara masuk melalui sisi belakang. Apabila seorang penerima telah terlayani maka dia harus meninggalkan antrian melalui sisi depan.

Dengan bentuk yang demikian antrian dapat dikatakan sebagai struktur data yang bersifat **FIFO (First In First Out**) artinya pengantri yang datang paling awal akan keluar paling dahulu.

Dalam antrian terdapat 2 buah operasi data yaitu (1) operasi **enQueue** dan (2) operasi **deQueue**. Operasi **enQueue** adalah operasi untuk menambahkan sebuah data ke dalam antrian dan dilakukan pada pintu belakang, sedangkan operasi **deQueue** adalah operasi untuk mengambil (menghapus) data dari dalam antrian dan dilakukan pada pintu depan.

Dalam implementasinya, sebuah antrian pasti memiliki kapasitas atau batas daya tampung antrian. Dalam ilustrasi di atas, kapasitas antrian biasanya disesuaikan dengan banyaknya zakat yang diberikan. Apabila hanya tersedia zakat untuk 10 orang maka kapasitas antrian hanya akan menampung 10 orang, sehingga orang ke-11, ke-12, dan seterusnya pasti akan ditolak untuk masuk ke dalam antrian.



Gambar 7.2 Ilustrasi Antrian

Ada 2 kondisi yang menjadi batasan sebuah antrian yaitu :

1. Kondisi **Antrian penuh**.
   Kondisi antrian penuh adalah kondisi di mana nilai `belakang` = nilai `batasAntrian`. Apabila kondisi ini terjadi maka antrian tidak lagi dapat menerima pemasukan data baru (enqueue data baru)

2. Kondisi **Antrian kosong**.
   Kondisi antrian kosong adalah kondisi di mana nilai `belakang` = -1. Apabila kondisi ini terjadi maka kita tidak lagi dapat melakukan pengambilan/penghapusan data (dequeue data) dari antrian

Program antrian dapat dilihat pada program 7.2 berikut ini.

```
public class program_antrian
{  public static int N = 5;
   public static int belakang = -1;

   public static void ENQUEUE (String antrian[], String data)
   {  if (belakang == N-1) //jika antrian penuh
      {  System.out.println("maap, antrian penuh, ENQUEUE "
                  + data + "  tidak dapat dilakukan");
      }
      else //jika antrian tidak penuh
      {  belakang = belakang + 1;
         antrian[belakang] = data;
         System.out.println("ENQUEUE " + data + " berhasil..");
      }
   }
   public static String DEQUEUE (String antrian[])
   {  String hasil;
      if (belakang < 0 ) //jika antrian kosong
      {  hasil = "ANTRIAN KOSONG, DEQUEUE GAGAL DILAKUKAN";
      }
      else //jika antrian tidak kosong
      {  hasil = antrian[0];
         //----menggeser data kedua dst, maju selangkah ke depan
         for (int i=0; i<=belakang-1;  i++)
         {  antrian[i] = antrian[i+1];
         }
         belakang = belakang - 1;
      }
      return (hasil);
   }

   public static void lihatAntrian(String antrian[])
   {  System.out.println("----------------------------------------");
      System.out.print("ISI ANTRIAN : (depan)");
      for (int i=0;  i<=belakang;  i++)
      {  System.out.print("  " + antrian[i]);
      }
      System.out.println("  (belakang)");
      System.out.println("----------------------------------------");
   }
   public static void main (String[] args)
   {  String antrian[] = new String[5];
      ............;
      ............;
      ............;
```

```
        ..............;
    }
}
```

Program 7.2.

6.3. Latihan

Latihan 1 :

Tuliskan program 7.1 menggunakan textpad. Tambahkan pada bagian program utama operasi "push" berturut-turut :

```
push (tumpukan, "Buku A");
push (tumpukan, "Buku B");
push (tumpukan, "Buku C");
push (tumpukan, "Buku D");
bacaTumpukan(tumpukan);
```

Sekarang jalankan program diatas, dan amati yang terjadi. Buku apa sajakah yang dapat di *push* ke dalam tumpukan? Apakah buku A, B, C, D, keempatnya dapat dipush ke dalam tumupukan?

Amatilah posisi "atas". Menurut anda, setelah dipush nya keempat buku di ABCD, berada di posisi manakah "atas"? Catatlah dalam laporan anda.

Latihan 2:

Sekarang, lakukan lagi proses *push* untuk sederet data "buku" berikut ini ke dalam tumpukan:

```
push (tumpukan, "Buku E");
push (tumpukan, "Buku F");
push (tumpukan, "Buku G");
bacaTumpukan(tumpukan);
```

Sekarang jalankan program diatas, dan amati yang terjadi. "Buku" apa sajakah yang dapat di *push* ke dalam tumpukan? Adakah "Buku" yang gagal di *push* ke dalam tumpukan? Apa sebabnya? Mengapa bisa demikian? Menurut anda bagian manakah dari program di atas yang menyebabkannya? Jelaskanlah dalam laporan anda.

Latihan 3:

Sekarang, lakukan *pop* dari tumpukan dengan cara menambahkan perintah berikut ke **bagian akhir** dari program utama .

```
System.out.println("POP: " + pop(tumpukan));
bacaTumpukan(tumpukan);
```

"Buku" apakah yang ter-*pop* ? Mengapa bisa demikian?

Sekarang amatilah kondisi tumpukan saat ini. Adakah yang berkurang? Mengapa hal itu bisa terjadi? Menurut anda, bagian manakah dari program di atas yang menyebabkan itu terjadi? Bagaimanakah juga dengan kondisi "atas" saat ini? Jelaskan pada laporan anda.

Latihan 4:

Sekarang, lakukan lagi proses *pop* dari tumpukan sebanyak 2 kali berturut-turut :

```
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
bacaTumpukan(tumpukan);
```

Catatlah "Buku" apakah saja yang ter-'*pop'* ?

Sekarang amatilah kondisi tumpukan saat ini. Adakah perubahan yang terjadi pada kondisi tumpukan saat ini dibanding dengan pada percobaan sebelumnya? Mengapa hal itu bisa terjadi? Bagaimanakah juga dengan kondisi "atas" saat ini? Jelaskan pada laporan anda.

Latihan 5:

Sekarang, lakukan lagi proses *pop* dari tumpukan sebanyak 3 kali berturut-turut :

```
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
bacaTumpukan(tumpukan);
```

Adakah proses *pop* yang tidak dapat (gagal) dilakukan? Mengapa bisa demikian? Jelaskan pada laporan anda.

Latihan 6 :

Tuliskan program 7.2 menggunakan textpad. Tambahkan pada bagian program utama operasi “enQueue” berturut-turut :

```
enQueue (antrian, "Mobil A");
enQueue (antrian, "Mobil B");
enQueue (antrian, "Mobil C");
bacaAntrian(antrian);
```

Sekarang jalankan program diatas, dan amati yang terjadi. “Mobil” apa sajakah yang dapat di *enQueue* ke dalam antrian? Lalu bagaimanakah dengan ‘belakang’? Berada di posisi manakah ‘belakang’ saat ini? Catatlah dalam laporan anda

Latihan 7:

Sekarang, lakukan lagi proses *enQueue* sederet data “Mobil” berikut ini ke dalam antrian:

```
enQueue (antrian, "Mobil D");
enQueue (antrian, "Mobil E");
enQueue (antrian, "Mobil F");
bacaAntrian(antrian);
```

Sekarang jalankan program diatas, dan amati yang terjadi. Bagaimanakah kondisi antrian saat ini? “Mobil” apa sajakah yang ada dalam antrian? Adakah “Mobil” yang gagal di *enQueue* ke dalam antrian? Apa sebabnya? Mengapa bisa demikian? Menurut anda bagian manakah dari program di atas yang menyebabkannya? Jelaskanlah dalam laporan anda.

Lalu bagaimana dengan posisi “belakang” saat ini? Adakah perbedaan posisi “belakang” saat ini jika dibandingkan dengan posisi “belakang” pada Latihan -6. Catatlah dalam laporan anda.

Latihan 8:

Sekarang, lakukan *deQueue* dari antrian dengan cara menambahkan perintah berikut ke bagian akhir dari program utama .

```
System.out.println("deQueue: " + deQueue(antrian));
bacaAntrian(antrian);
```

“Mobil” apakah yang ter-*deQueue* ? Mengapa bisa demikian?

Sekarang amatilah kondisi antrian saat ini. Adakah perubahan yang terjadi pada kondisi antrian saat ini dibanding dengan pada percobaan sebelumnya? Apakah setiap data “Mobil B,C,D,..dst” masih berada di posisinya yang sama? Mengapa hal itu bisa terjadi? Menurut anda, bagian manakah dari program di atas yang menyebabkan itu terjadi? Bagaimanakah juga dengan kondisi ekor saat ini? Jelaskan pada laporan anda.

Latihan 9:

Sekarang, lakukan lagi proses *deQueue* dari antrian sebanyak 3 kali berturut-turut :

```
System.out.println("deQueue: " + deQueue(antrian));
System.out.println("deQueue: " + deQueue(antrian));
System.out.println("deQueue: " + deQueue(antrian));
bacaAntrian(antrian);
```

Catatlah “Mobil” apakah saja yang ter-*deQueue* ?

Sekarang amatilah kondisi antrian saat ini. Adakah perubahan yang terjadi pada kondisi antrian saat ini dibanding dengan pada percobaan sebelumnya? Apakah setiap data “Mobil B,C,D,..dst” masih berada di posisinya yang sama? Mengapa hal itu bisa terjadi? Bagaimanakah juga dengan kondisi ekor saat ini? Jelaskan pada laporan anda.

Latihan 10:

Sekarang, lakukan lagi proses *deQueue* dari antrian sebanyak 2 kali berturut-turut.

```
System.out.println("deQueue: " + deQueue(antrian));
System.out.println("deQueue: " + deQueue(antrian));
bacaAntrian(antrian);
```

Adakah proses *deQueue* yang tidak dapat dilakukan? Mengapa bisa demikian? Jelaskan pada laporan anda.

Latihan
Tugas