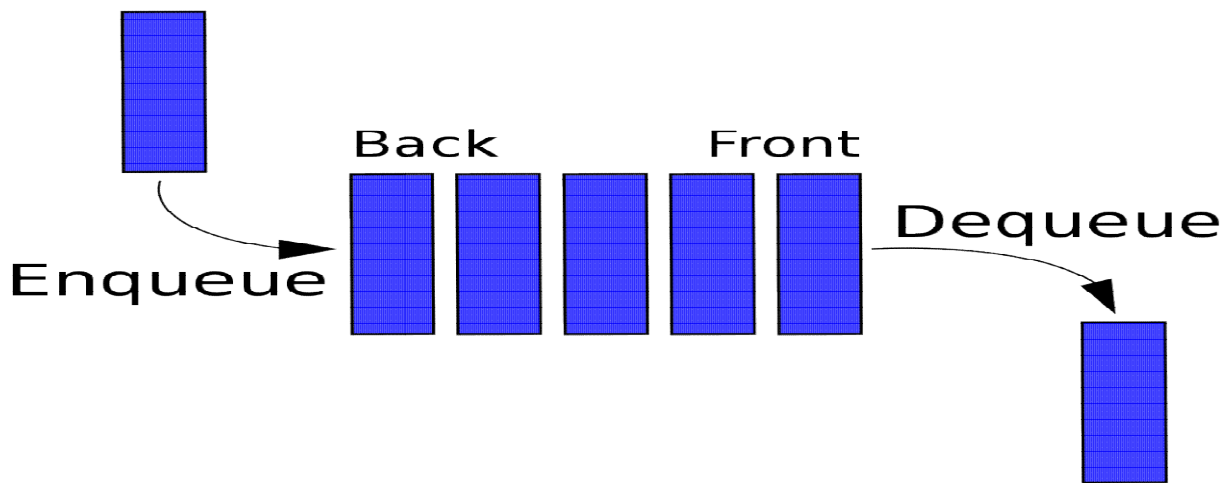


Program Antrian Java (OPERASI QUEUE)



Pertama, buatlah class **JavaQueue** lengkap dengan Main methodnya.

```
public class JavaQueue {  
    public static void main(String[] args) {  
    }  
}
```

Kedua, import: `java.util.Scanner` dan `java.util.InputMismatchException`. Letakkan di atas nama class

```
import java.util.Scanner;  
import java.util.InputMismatchException;
```

Ketiga, buatlah dua variabel. Variabel pertama adalah Integer Array dengan panjang 5 dengan nama *queue*, dan yang kedua adalah variabel Integer dengan nama variabel *counters* dan dengan nilai awal = 0. Variabel Integer Array inilah storage yang Saya gunakan untuk *queue* Saya dan sebagai pointer dan counter-nya, Saya menggunakan variabel *counters*. Letakkan kedua variabel setelah nama class.

```
private static int[] queue = new int[5];  
private static int counters = 0;
```

Keempat, buat sebuah return value method untuk memeriksa tingkat keterisian antrian yang mana akan mengembalikan nilai `TRUE` jika masih terdapat tempat dalam antrian, dan memberikan nilai `FALSE` jika antrian sudah penuh.

```
private static boolean queueStorage() {
```

```

        if(counters < queue.length) {
            return true;
        }
        else {
            return false;
        }
    }
}

```

Kelima, buat sebuah method void untuk melakukan aksi *Enqueue*.

```

private static void createQueue() {
    int loopX = 0;
    int alpha = 0;
    while(loopX == 0) {
        System.out.print("Masukkan Data (angka): ");
        Scanner alphaX = new Scanner(System.in);
        try {
            alpha = alphaX.nextInt();
            loopX = 1;
        }
        catch(InputMismatchException e) {
            System.out.println("Masukan harus berupa Angka!");
            loopX = 0;
        }
    }
    queue[counters] = alpha;
    counters++;
}

```

Mari kita lihat sejenak method di atas. Karena saya ingin menghandle inputan agar benar – benar terbebas dari error, maka terlebih dahulu saya pastikan bahwa proses pemasukan (input) value adalah benar. Jika tidak, maka proses pemasukan (input) value akan terus diulang hingga memperoleh masukan yang benar; yakni berupa angka. Oleh karena itu, Saya buat variabel integer loopX dengan nilai awal 0; Apabila variabel loopX ini masih terus menerus bernilai 0, maka proses pemasukan (input) akan terus diulang.

Kemudian, saya buat variabel lokal integer dengan nama alpha dan dengan nilai awal 0 sebagai variabel untuk menampung masukan.

Masuk ke dalam looping while, sudah Saya jelaskan tadi, bahwa selama nilai loopX adalah 0, maka looping akan terus dilakukan: hingga mendapatkan input yang benar. Oleh karena itu, untuk memastikan bahwa inputannya benar, Saya handle inputan Scanner dengan menggunakan try catch. Ketika try, dan memperoleh inputan yang benar, maka loopX akan bernilai 1, dan looping selesai. Namun ketika try, ternyata memperoleh inputan yang salah (inputan selain angka), maka selain menampilkan pesan error, looping juga akan terus dilakukan, hingga memperoleh inputan yang benar.

Setelah keluar dari looping, inputan dimasukkan ke dalam array queue sebagai antrian dan value counters bertambah 1. Yang artinya, sudah ada 1 value di dalam antrian. Counters akan terus bertambah hingga batas titik maksimal array queue dapat menampung (5 value).

Keenam, buat sebuah method void untuk melakukan aksi *Dequeue*.

```
private static void removeQueue() {
    counters--;
    for(int i = 0; i < counters; i++) {
        queue[i] = queue[i + 1];
    }
    System.out.println("Data pertama dalam queue sudah dikeluarkan");
}
```

Mari kita perhatikan kembali blok kode di atas. Logika *dequeue* adalah, menghapus value di kepala antrian, dan menggantinya dengan value di belakangnya, dan terus dilakukan hingga di ekor antrian. Dengan melakukan *dequeue*, artinya kita juga mengurangi total antrian. Misal tadinya 5 menjadi 4. Oleh karena itu, di awal method, Saya kurangi terlebih dahulu nilai *counters* dengan melakukan decrements sehingga nilai *counters* berkurang 1. Setelahnya, dengan menggunakan nilai *counters* yang baru, Saya melakukan looping dengan menggunakan for untuk memindahkan value – value dalam queue ke depan (1 ke 0, 2 ke 1, 3 ke 2, 4 ke 3). Setelah melakukan looping, otomatis kolom antrian urutan ke 4 menjadi kosong (0, 1, 2, 3, 4) dan dapat ditempati.

Ketujuh, untuk dapat membuktikan bahwa queue bekerja dengan baik, Saya menambahkan satu method lagi untuk menampilkan nilai – nilai dalam *queue*.

```
private static void displayDataQueue() {
    System.out.print("Data dalam Queue: ");
    for(int i = 0; i < counters; i++) {
        System.out.print(" ["+i+" => "+queue[i]+"]" );
    }
    System.out.println("");
}
```

Secara sederhana, dengan menggunakan looping, data – data dalam *queue* ditampilkan

Kedelapan, Saya buat juga satu method untuk menghapus seluruh antrian.

```
private static void cleanQueue() {
    counters = 0;
}
```

Dengan men-set value *counters* menjadi 0 sama saja dengan me-reset pointer dan total value yang ada di dalam array *queue*.

Selanjutnya, Saya membuat menu untuk program ini. Oleh karena itu, Saya menambahkan satu method lagi untuk menampilkan menu.

```
private static void menuProgram() {
    int loopX = 0;
    int choosenMenu = 0;
    while(loopX == 0) {
        System.out.println("\nContoh Program Queue dengan Java");
    }
}
```

```

        System.out.println("Menu: ");
        System.out.println("1. Tambah Queue");
        System.out.println("2. Keluarkan 1 data dari Queue");
        System.out.println("3. Status Queue");
        System.out.println("4. Tampilkan data dalam Queue");
        System.out.println("5. Bersihkan Queue");
        System.out.println("6. Keluar dari Program");
        System.out.print("Pilihan Menu (1 - 6) >>> ");
        Scanner menuOption = new Scanner(System.in);
        try {
            choosenMenu = menuOption.nextInt();
            loopX = 1;
        } catch (InputMismatchException e) {
            System.out.println("Masukan harus Angka!");
        }
    }
    System.out.println("");
    menuChooser(choosenMenu);
}

```

Untuk menghandle item menu yang dipilih, Saya buat juga membuat satu method lagi.

```

private static void menuChooser(int choosenMenu) {
    switch(choosenMenu) {
        case 1:
            boolean check = queueStorage();
            if(check) {
                createQueue();
            }
            else {
                System.out.println("Antrian Penuh!, kosongkan data satu
terlebih dahulu!");
            }
            break;
        case 2:
            removeQueue();
            break;
        case 3:
            System.out.println("Status Storage: ");
            System.out.println("Kapasitas: " + queue.length);
            System.out.println("Terisi    : " + counters);
            break;
        case 4:
            displayDataQueue();
            break;
        case 5:
            cleanQueue();
            break;
        case 6:
            quitApp();
            break;
    }
    menuProgram();
}

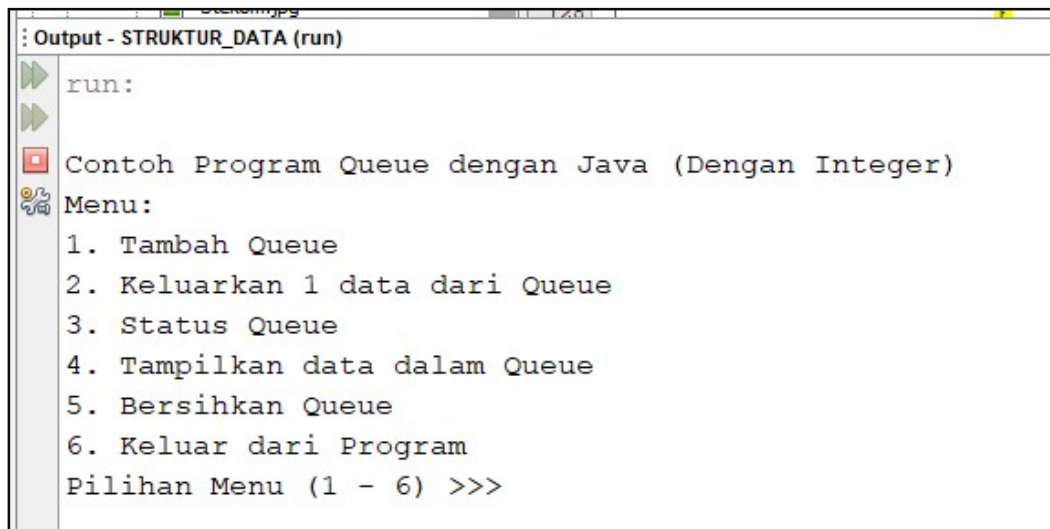
```

Karena Saya menawarkan fungsi exit dari program, maka Saya juga menambahkan satu method untuk men-terminate program.

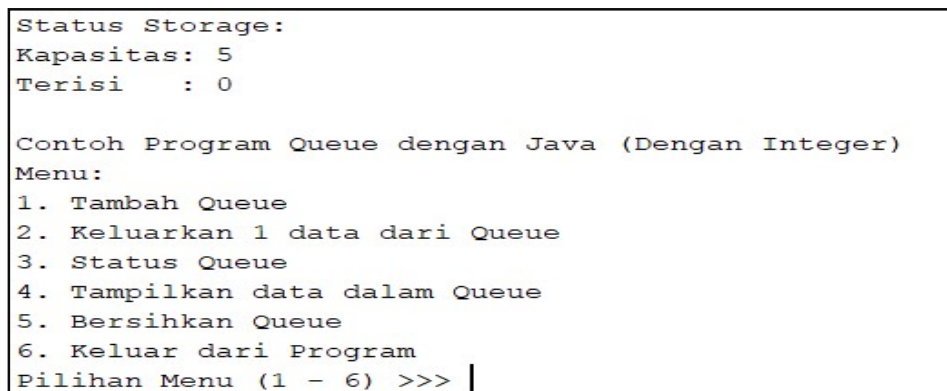
```
private static void quitApp() {  
    String quitss = "y";  
    System.out.print("Keluar dari Program? (Y/T): ");  
    quitss = new Scanner(System.in).nextLine();  
    if(quitss.equalsIgnoreCase("y")) {  
        System.exit(0);  
    }  
    else {  
        menuProgram();  
    }  
}
```

Terakhir, Saya panggil method menuProgram dari method Main.

```
public static void main(String[] args) {  
    menuProgram();  
}
```



```
run:  
Contoh Program Queue dengan Java (Dengan Integer)  
Menu:  
1. Tambah Queue  
2. Keluarkan 1 data dari Queue  
3. Status Queue  
4. Tampilkan data dalam Queue  
5. Bersihkan Queue  
6. Keluar dari Program  
Pilihan Menu (1 - 6) >>>
```



```
Status Storage:  
Kapasitas: 5  
Terisi : 0  
  
Contoh Program Queue dengan Java (Dengan Integer)  
Menu:  
1. Tambah Queue  
2. Keluarkan 1 data dari Queue  
3. Status Queue  
4. Tampilkan data dalam Queue  
5. Bersihkan Queue  
6. Keluar dari Program  
Pilihan Menu (1 - 6) >>> |
```