# LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

Dosen Pengajar : Triana Fatmawati, S.T, M.T

# Jobsheet-16 Collection



Nama : Surya Rahmat Fatahillah

NIM : 2341760020

Prodi : Sistem Informasi Bisnis

# JURUSAN TEKNOLOGI INFORMASI POLITEKNIK NEGERI MALANG 2023/2024

# 16.2.1. Percobaan 1

# Pada percobaan 1 ini akan dicontohkan penggunaan collection untuk menambahkan sebuah

# elemen, mengakses elemen, dan menghapus sebuah elemen.

# Buatlah sebuah class ContohList yang main methode berisi kode program seperti di bawah ini

# 

# Tambahkan kode program untuk menggunakan collection dengan aturan penulisan kode program seperti berikut

# 

# 16.2.2. Verifikasi Hasil Percobaan

# Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

# 

**16.2. Kegiatan Praktikum 1**

**16.2.3. Pertanyaan Percobaan**

1. Perhatikan baris kode 25-36, mengapa semua jenis data bisa ditampung ke dalam sebuah

Arraylist?

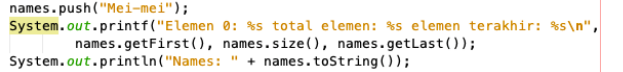
2. Modifikasi baris kode 25-36 seingga data yang ditampung hanya satu jenis atau spesifik tipe

tertentu!

3. Ubah kode pada baris kode 38 menjadi seperti ini



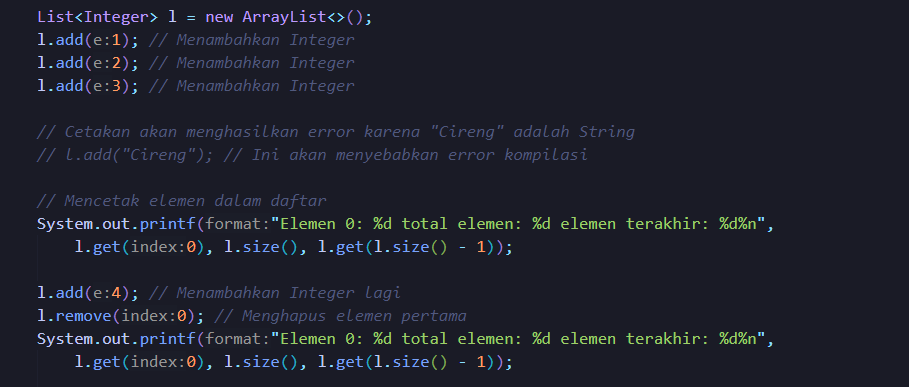
4. Tambahkan juga baris berikut ini, untuk memberikan perbedaan dari tampilan yang sebelumnya



5. Dari penambahan kode tersebut, silakan dijalankan dan apakah yang dapat Anda jelaskan!

**Jawaban!**

1. Hal ini dikarenakan ArrayList bisa menampung berbagai jenis data berasal dari penggunaan tipe generic dan sifat kelas Object sebagai superclass dari semua kelas di Java. ArrayList di Java adalah bagian dari Java Collections Framework dan menggunakan konsep generics, yang memungkinkan ArrayList untuk menampung berbagai jenis data. Saat kita mendeklarasikan sebuah ArrayList tanpa menentukan tipe data spesifik. Ini berarti ArrayList tersebut dapat menampung objek dari tipe apa pun karena tipe Object adalah superclass dari semua kelas di Java.
2. Berikut modifikasi pada baris kode 25-36 sehingga data yang ditampung hanya memiliki satu jenis tipe data



Ouput:

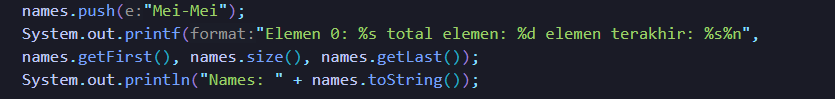


Untuk memodifikasi ArrayList supaya hanya dapat menampung satu jenis data atau tipe spesifik, Kita harus menggunakan generics dengan tipe tertentu saat mendeklarasikan ArrayList. Misalnya, jika kita ingin ArrayList hanya menampung Integer, kita dapat mendeklarasikan ArrayList<Integer>. Begitu juga jika kita ingin menampung String saja, kita dapat menggunakan ArrayList<String>.

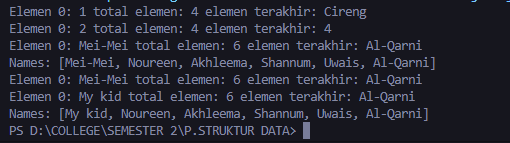
1. Berikut kode baris 38 setelah di ubah



1. Berikut penambahan kode nya



1. Kemudian berikut merupakan hasilnya:



Dengan mengubah deklarasi List<String> names = new LinkedList<>(); menjadi LinkedList<String> names = new LinkedList<>(); dan menambahkan names.push("Mei-Mei"); serta menggunakan getFirst() dan getLast(), kita mengakses elemen pertama dan terakhir dalam LinkedList.

Berikut penjelasannya:

* push("Mei-Mei"): Ini akan memasukkan elemen baru "Mei-Mei" ke dalam LinkedList. Fungsi push() khusus untuk LinkedList dan menambahkan elemen ke awal daftar.
* getFirst(): Metode ini mengembalikan elemen pertama dalam LinkedList.
* getLast(): Metode ini mengembalikan elemen terakhir dalam LinkedList.

Kemudian cetakan akan menampilkan elemen pertama, jumlah total elemen dalam LinkedList, dan elemen terakhir.

Setelah itu, kita mencetak ulang LinkedList dengan menggunakan toString() untuk melihat semua elemennya setelah penambahan.

**16.3. Kegiatan Praktikum 2**

**16.3.1. Tahapan Percobaan**

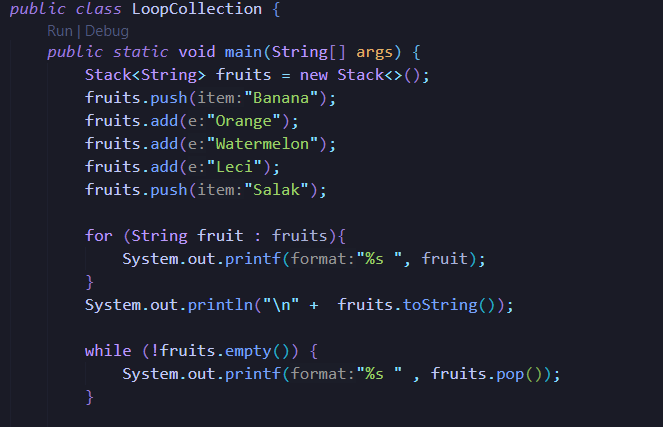
Pada praktikum 2 ini akan dibuat beberapa method untuk menampilkan beberapa cara yang

dapat dilakukan untuk mengambil/menampilkan elemen pada sebuah collection. Silakan ikutilah

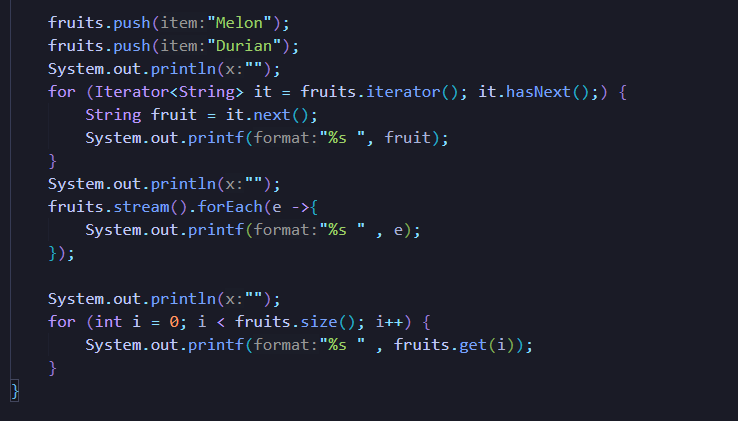
Langkah-langkah di bawah ini

1. Buatlah class dengan nama LoopCollection serta tambahkan method main yang isinya adalah

sebagai berikut.

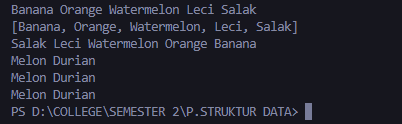


2. Tambahkan potongan kode berikut ini dari yang sebelumnya agar proses menampilkan elemen pada sebuah stack bervariasi



**16.3.2. Verifikasi Hasil Percobaan**

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini



**16.3.3. Pertanyaan Percobaan**

1. Apakah perbedaan fungsi push() dan add() pada objek fruits?

2. Silakan hilangkan baris 43 dan 44, apakah yang akan terjadi? Mengapa bisa demikian?

3. Jelaskan fungsi dari baris 46-49?

4. Silakan ganti baris kode 25, Stack<String> menjadi List<String> dan apakah yang terjadi?

Mengapa bisa demikian?

5. Ganti elemen terakhir dari dari objek fruits menjadi “Strawberry”!

6. Tambahkan 3 buah seperti “Mango”,”guava”, dan “avocado” kemudian dilakukan sorting!

**Jawaban!**

1. Pada dasarnya, perbedaan antara push() dan add() pada objek Stack adalah sebagai berikut:

* push(): Metode push() digunakan khusus untuk menambahkan elemen ke tumpukan (stack). Ini menempatkan elemen baru di bagian atas tumpukan.
* add(): Metode add() adalah metode turunan dari kelas Vector, dan ia menambahkan elemen ke tumpukan di posisi tertentu. Namun, karena tumpukan biasanya hanya menambahkan elemen di bagian atasnya, add() biasanya tidak direkomendasikan untuk digunakan pada objek Stack.

1. Berikut merupakan output program ketika baris 43 dan 44



Hal ini terjadi karena tidak ada program yang melakukan push dengan nama **Melon Durian**, yang mana ketika tidak ada push dengan nama tersebut maka iterasi pencetakan program tidak akan dilakukan.

1. Di baris pertama, **fruits.iterator()** mengembalikan iterator yang digunakan untuk mengakses elemen-elemen dalam tumpukan. Kemudian, dalam loop while, **it.hasNext()** digunakan untuk memeriksa apakah masih ada elemen yang tersisa dalam tumpukan untuk diakses. Jika masih ada, **it.next()** digunakan untuk mendapatkan elemen berikutnya dalam tumpukan.

Dalam setiap iterasi, nilai elemen yang didapatkan dari iterator disimpan dalam variabel fruit, dan kemudian dicetak menggunakan **System.out.printf("%s ", fruit);.**

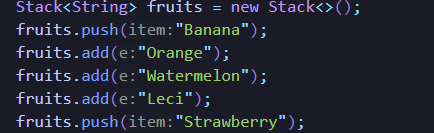
Jadi, secara keseluruhan, baris kode ini mencetak semua elemen yang ada dalam di dalam **fruits**, menggunakan iterator untuk mengakses elemen-elemen tersebut satu per satu.

1. Jika kita mengganti baris kode **Stack<String>** menjadi **List<String>,** maka akan terjadi kesalahan kompilasi. Hal ini terjadi karena **List** adalah sebuah antarmuka (interface) di Java, sedangkan Stack adalah sebuah kelas konkret. Kita tidak dapat langsung menginstansiasi sebuah antarmuka; kita harus menggunakan salah satu kelas yang mengimplementasikan antarmuka tersebut.

Jadi, Kita perlu memilih salah satu kelas yang mengimplementasikan antarmuka List, seperti ArrayList, LinkedList, atau Vector. Misalnya, jika Kita ingin menggunakan ArrayList, baris kode 25 akan menjadi:



1. Berikut kode program ketika sudah mengganti elemen terakhir dari dari objek fruits menjadi “Strawberry”

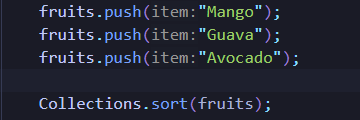


1. Berikut kode program yang sudah dimodifikasi dengan menambahkan 3 buah seperti “Mango”,”guava”, dan “avocado” kemudian dilakukan sorting

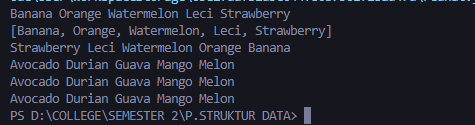
* Pertama kita menambahkan library collection



* Kemudian tambahkan kode berikut untuk menambahkan data dan melakukan sorting



* Kemudian berikut hasilnya:

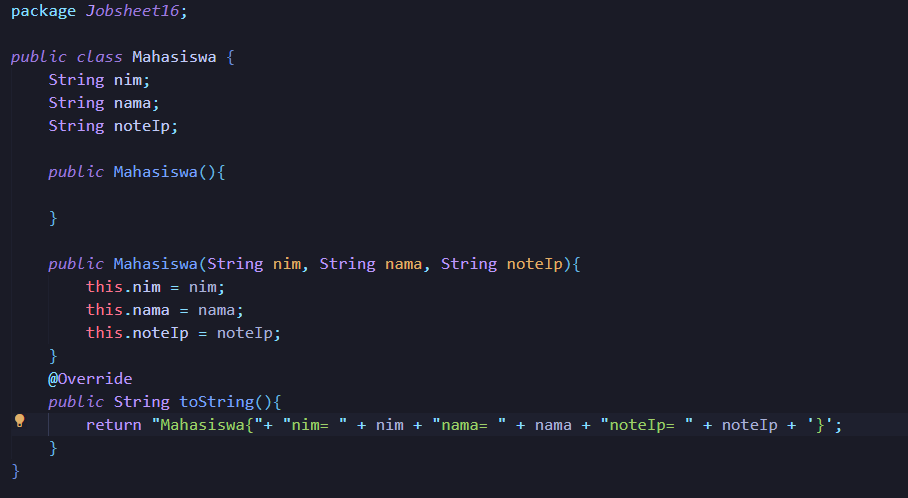


**16.4. Kegiatan Praktikum 3**

**16.4.1. Tahapan Percobaan**

Pada praktikum 3 ini dilakukan uji coba untuk mengimplementasikan sebuah collection untuk menampung objek yang dibuat sesuai kebutuhan. Objek tersebut adalah sebuah objek mahasiswa dengan fungsi-fungsi umum seperti menambahkan, menghapus, mengubah, dan mencari

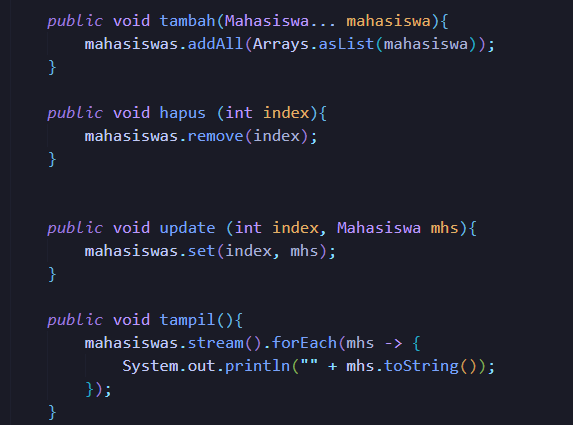
1. Buatlah sebuah class Mahasiswa dengan attribute, kontruktor, dan fungsi sebagai berikut.



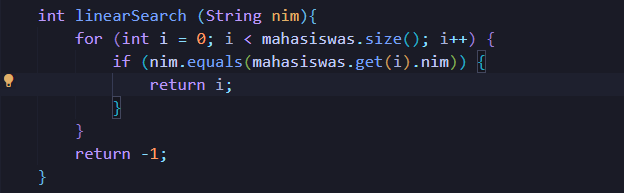
1. Selanjutnya, buatlah sebuah class ListMahasiswa yang memiliki attribute seperti di bawah ini



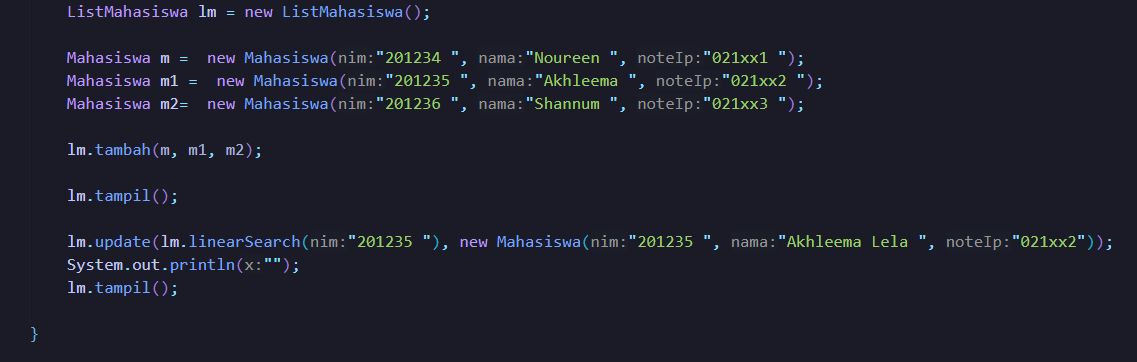
1. Method tambah(), hapus(), update(), dan tampil()secara berurut dibuat agar bisa melakukan operasi-operasi seperti yang telah disebutkan.



1. Untuk proses hapus, update membutuhkan fungsi pencarian terlebih dahulu yang potongan kode programnya adalah sebagai berikut.

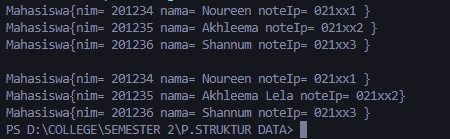


1. Pada class yang sama, tambahkan main method seperti potongan program berikut dan amati hasilnya!



**16.4.2. Verifikasi Hasil Percobaan**

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini



**16.4.3. Pertanyaan Percobaan**

1. Pada fungsi tambah() yang menggunakan unlimited argument itu menggunakan konsep apa?

Dan kelebihannya apa?

2. Pada fungsi linearSearch() di atas, silakan diganti dengan fungsi binarySearch() dari collection!

3. Tambahkan fungsi sorting baik secara ascending ataupun descending pada class tersebut!

**Jawaban!**

1. Fungsi **tambah()** yang menggunakan parameter varargs (Mahasiswa... mahasiswa) menerapkan konsep varargs (variadic arguments). Varargs memungkinkan Kita untuk meneruskan jumlah argumen yang bervariasi ke suatu metode atau fungsi.

Kelebihan dari menggunakan varargs adalah fleksibilitas dalam penggunaannya. Dengan varargs, Kita dapat memanggil metode dengan jumlah argumen yang berbeda-beda tanpa harus membuat berbagai versi dari metode tersebut. Hal ini sangat berguna ketika Kita tidak tahu pasti berapa banyak argumen yang akan Kita lewatkan.

Dalam kasus tambah() di kelas ListMahasiswa, kita menggunakan varargs yang memungkinkan kita untuk menambahkan sejumlah Mahasiswa ke dalam list dengan memanggil metode tambah() dengan hanya sekali panggil, tanpa perlu membuat overloading metode untuk menerima berbagai jumlah Mahasiswa.

1. Berikut modifikasi ketika fungsi linearSearch() diganti dengan fungsi binarySearch() dari collection

* Class Mahasiswa

Untuk menggunakan binarySearch() dari kelas Collections, elemen-elemen dalam list harus dapat dibandingkan satu sama lain.

Untuk mengaplikasikannya, Kita perlu mengimplementasikan antarmuka Comparable di kelas Mahasiswa dan menentukan kriteria perbandingan yang sesuai (misalnya, menggunakan NIM). Berikut adalah contoh kelas Mahasiswa yang diperbarui dengan implementasi antarmuka Comparable:

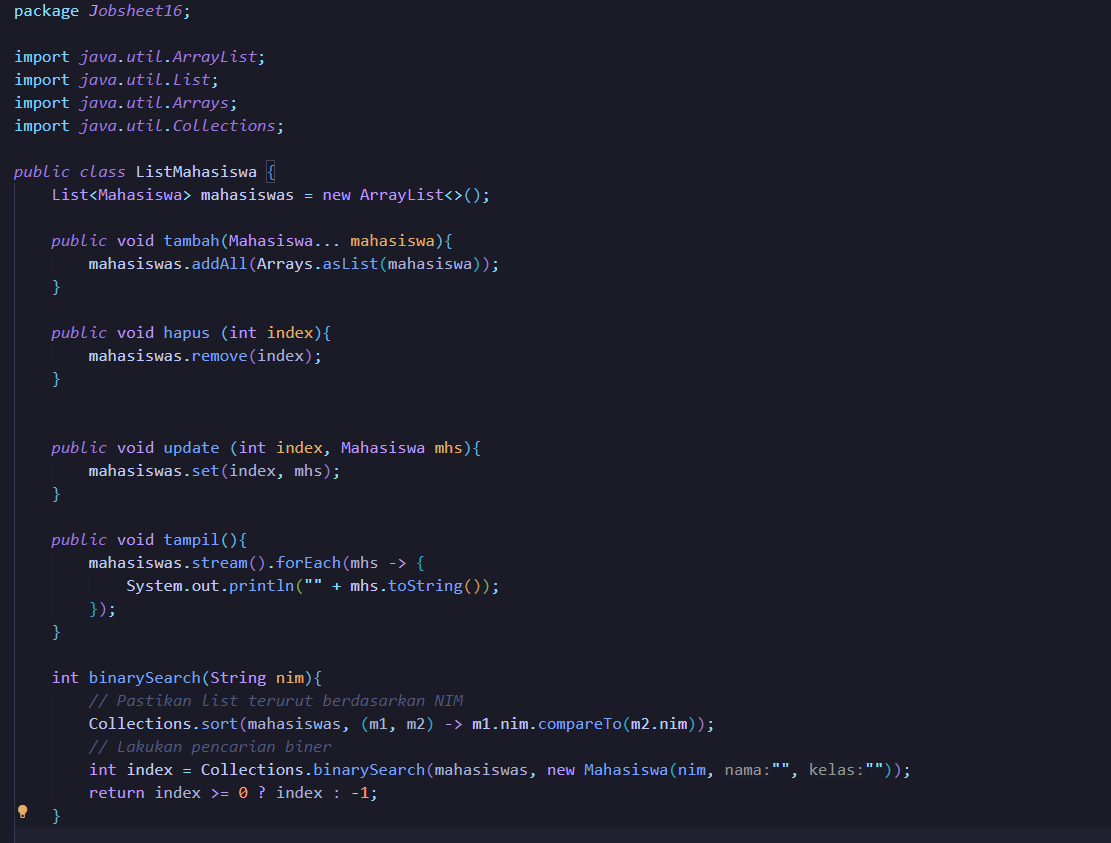


* Class List Mahasiswa

Untuk mengganti metode linearSearch() dengan binarySearch() dari kelas Collections, Anda harus memastikan bahwa elemen-elemen dalam list terurut secara teratur. binarySearch() bekerja dengan benar hanya pada list yang sudah terurut. Berikut adalah perubahan yang perlu Anda lakukan:

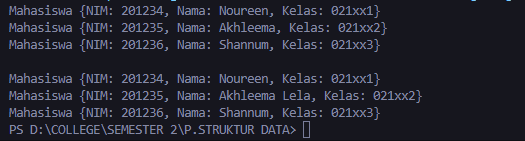
1. Pastikan bahwa elemen-elemen dalam mahasiswas sudah terurut berdasarkan kriteria yang Anda inginkan (misalnya, berdasarkan NIM).
2. Ganti pemanggilan binarySearch() pada metode main() dengan parameter yang sesuai.

Berikut adalah perubahan yang perlu Anda lakukan di kelas ListMahasiswa:



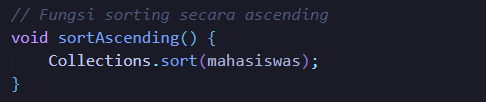


Kemudian berikut hasilnya:

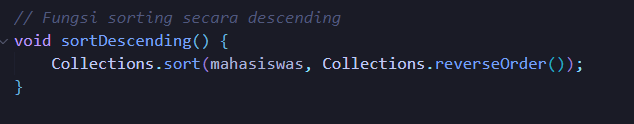


1. Berikut kode program dengan menambahkan fungsi sorting baik secara ascending ataupun descending pada class tersebut!

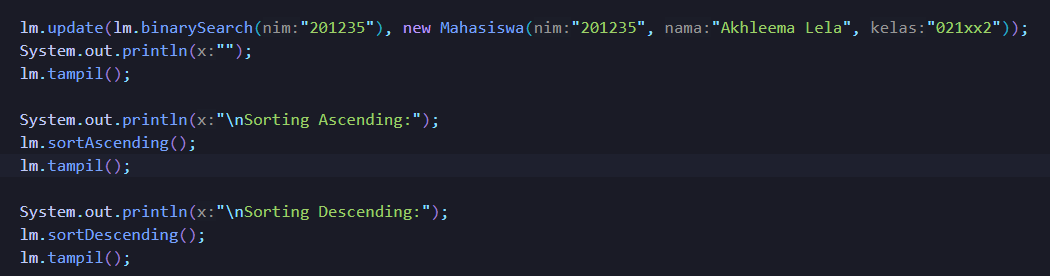
* Sort Ascending



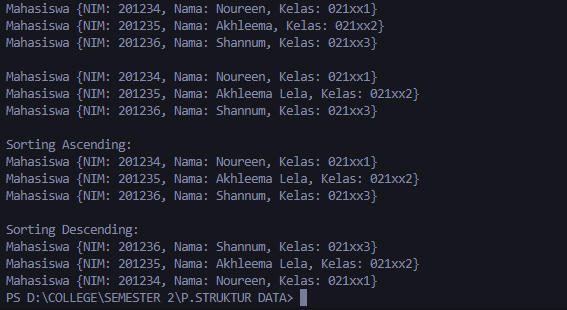
* Sort Descending



* Fungsi Main



Hasil Program:



**16.5. Tugas Praktikum**

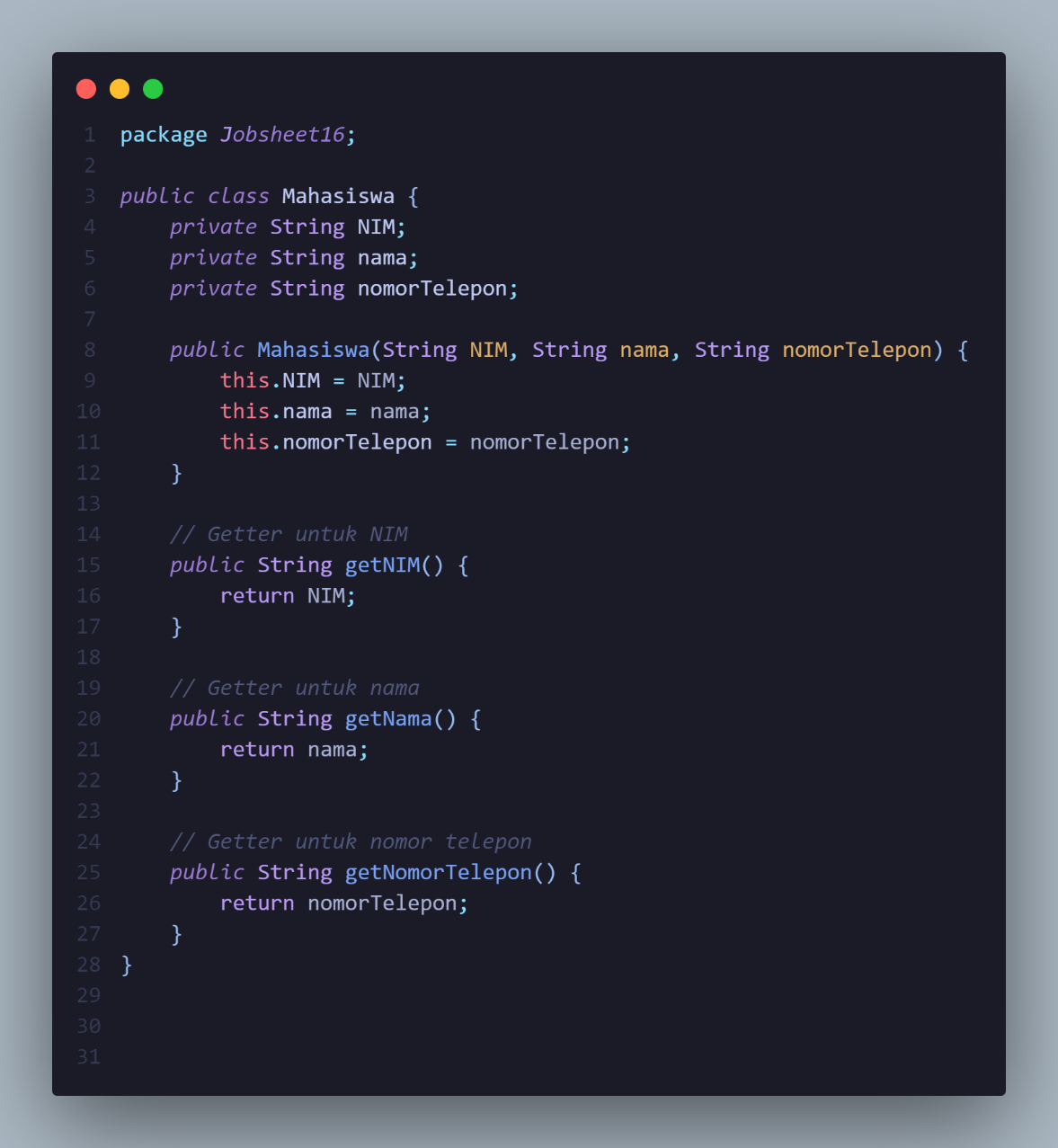
1. Buatlah implementasi program daftar nilai mahasiswa semester, minimal memiliki 3 class yaitu Mahasiswa, Nilai, dan Mata Kuliah. Data Mahasiswa dan Mata Kuliah perlu melalui penginputan data terlebih dahulu.

2. Tambahkan prosedur hapus data mahasiswa melalui implementasi Queue pada collections Tugas nomor 1!

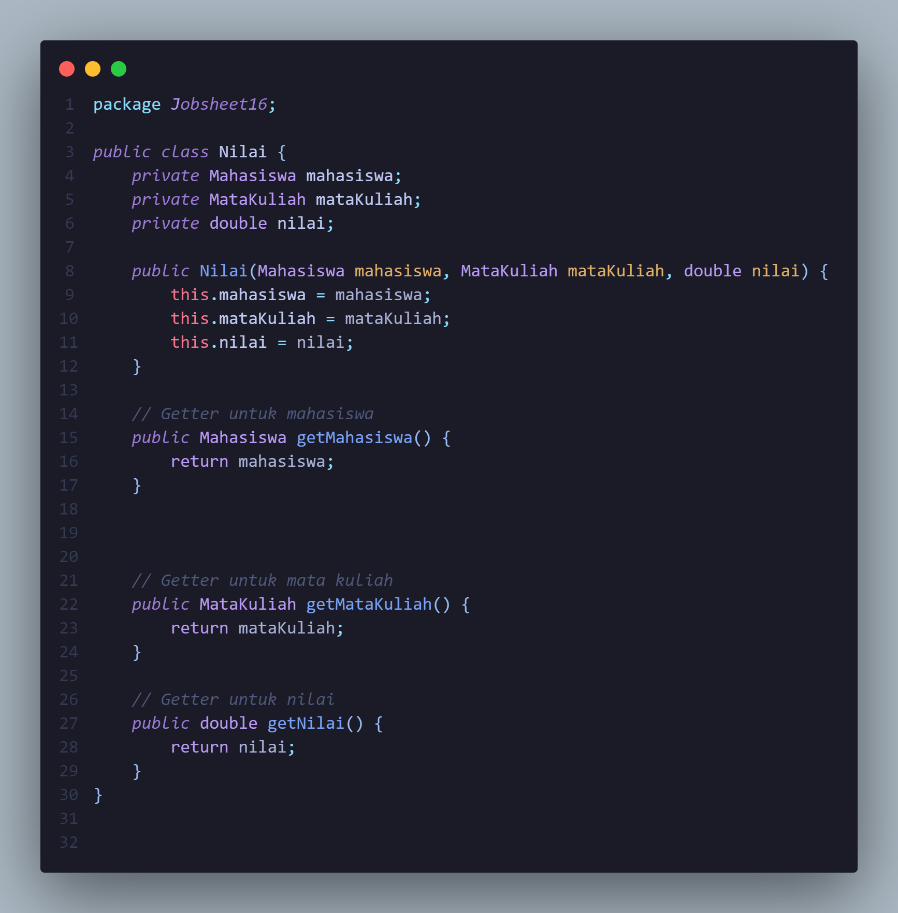
**Jawaban!**

1. Berikut merupakan program daftar nilai mahasiswa semester dengan 5 class yaitu Mahasiswa, Nilai, Mata Kuliah, Program dan Main.

* Class Mahasiswa



* Class Nilai



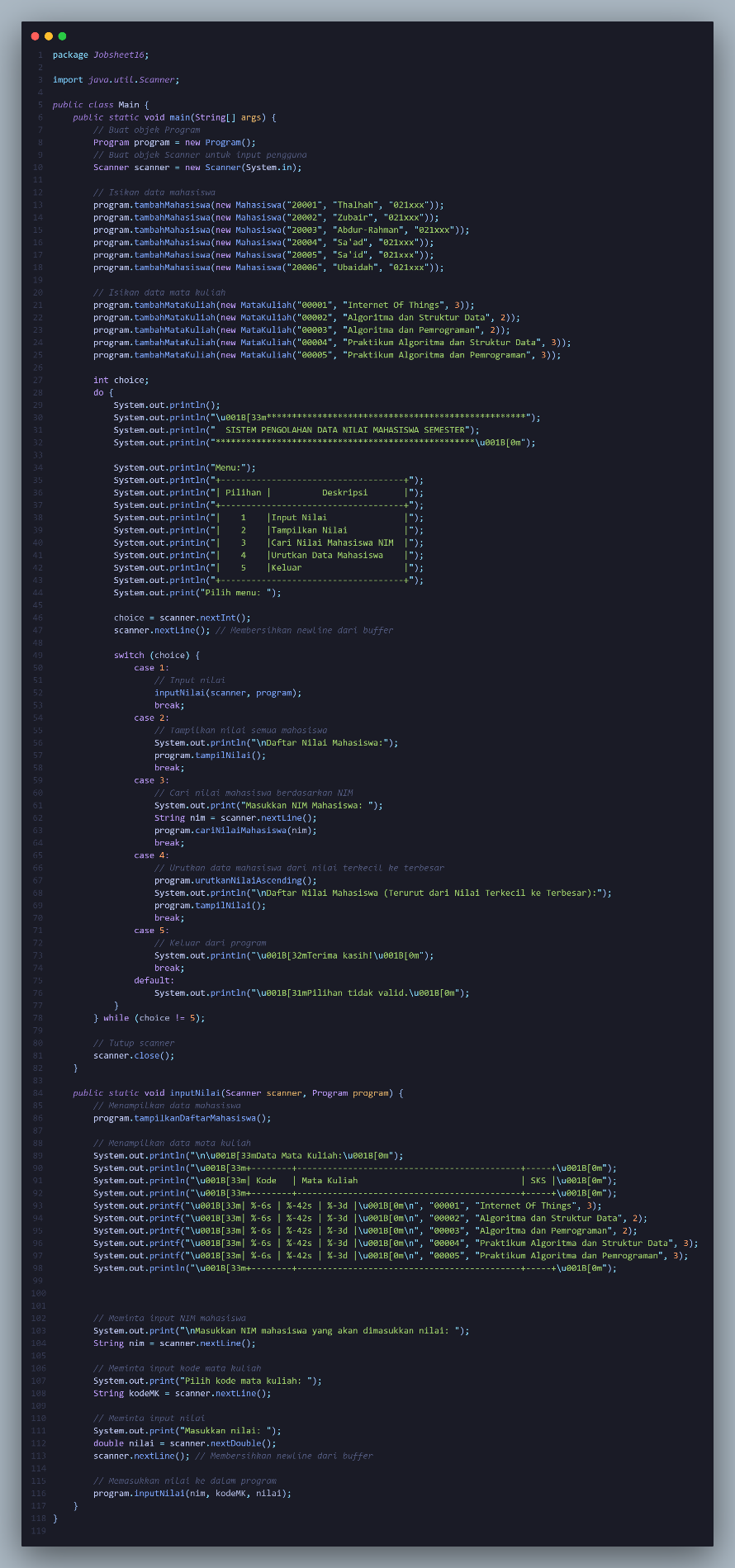
* Class Mata Kuliah



* Class Program



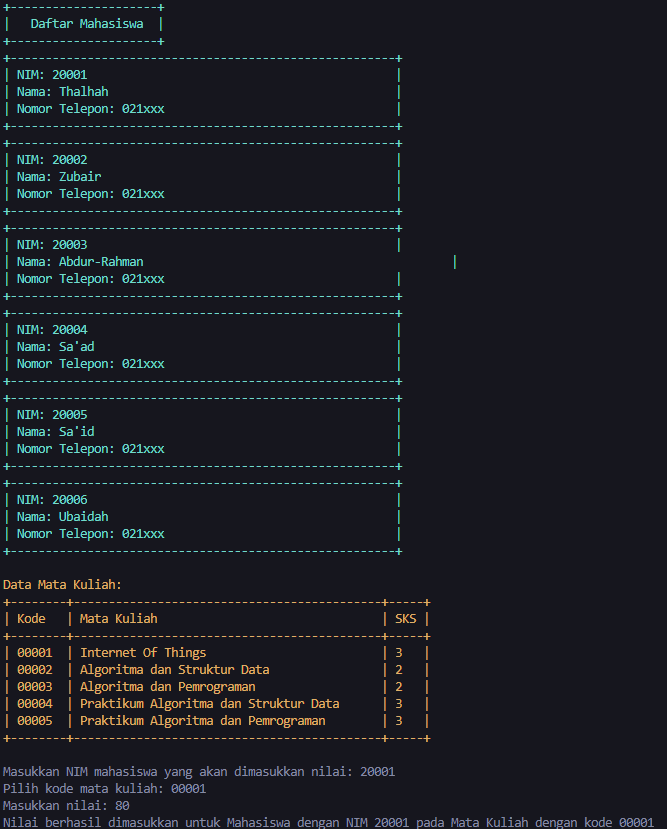
* Class Main



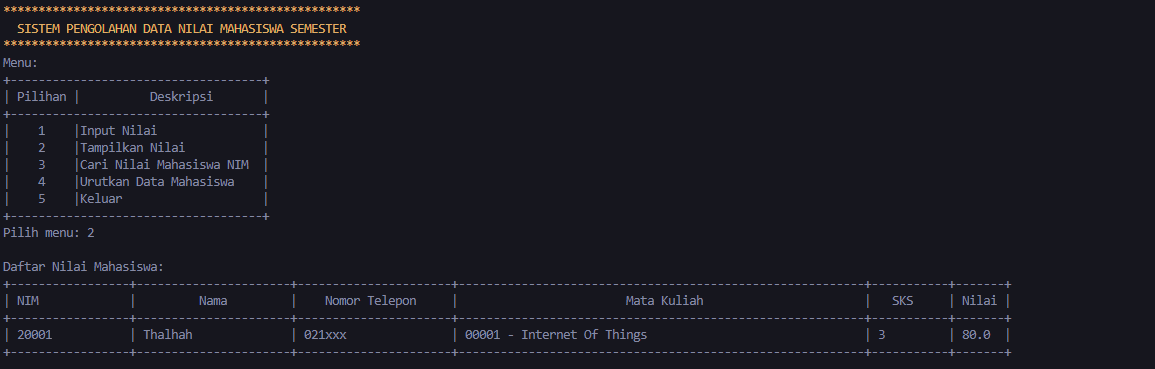
Berikut hasil program nya:

* Fitur Input Nilai

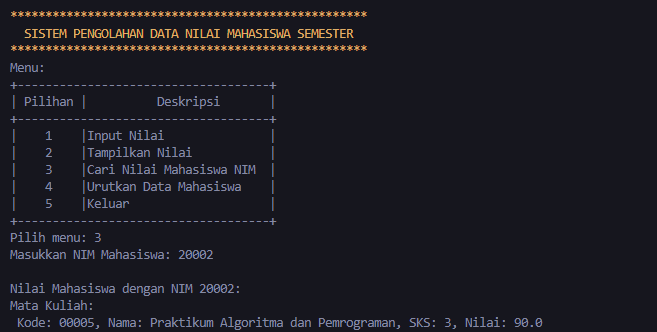




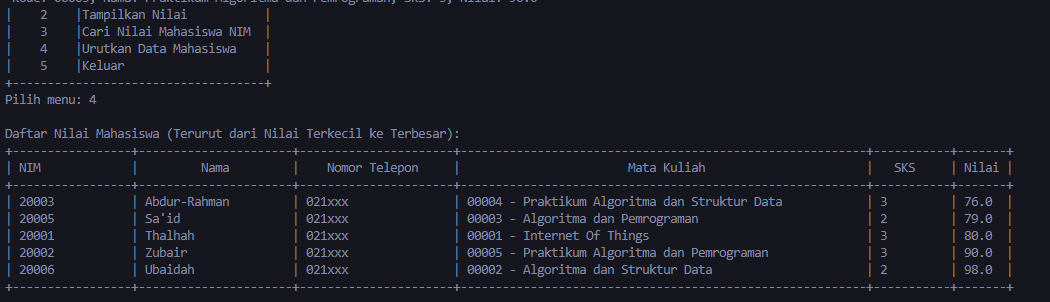
* Fitur Tampilkan Nilai



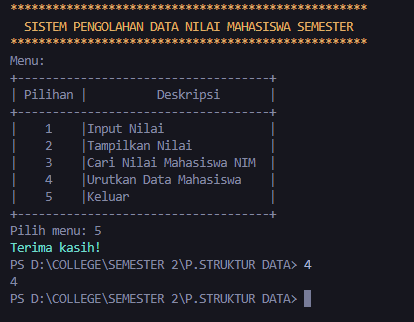
* Fitur Cari Nilai Berdasarkan NIM



* Fitur Urutkan Data Mahasiswa (Berdasarkan Nilai)



* Keluar



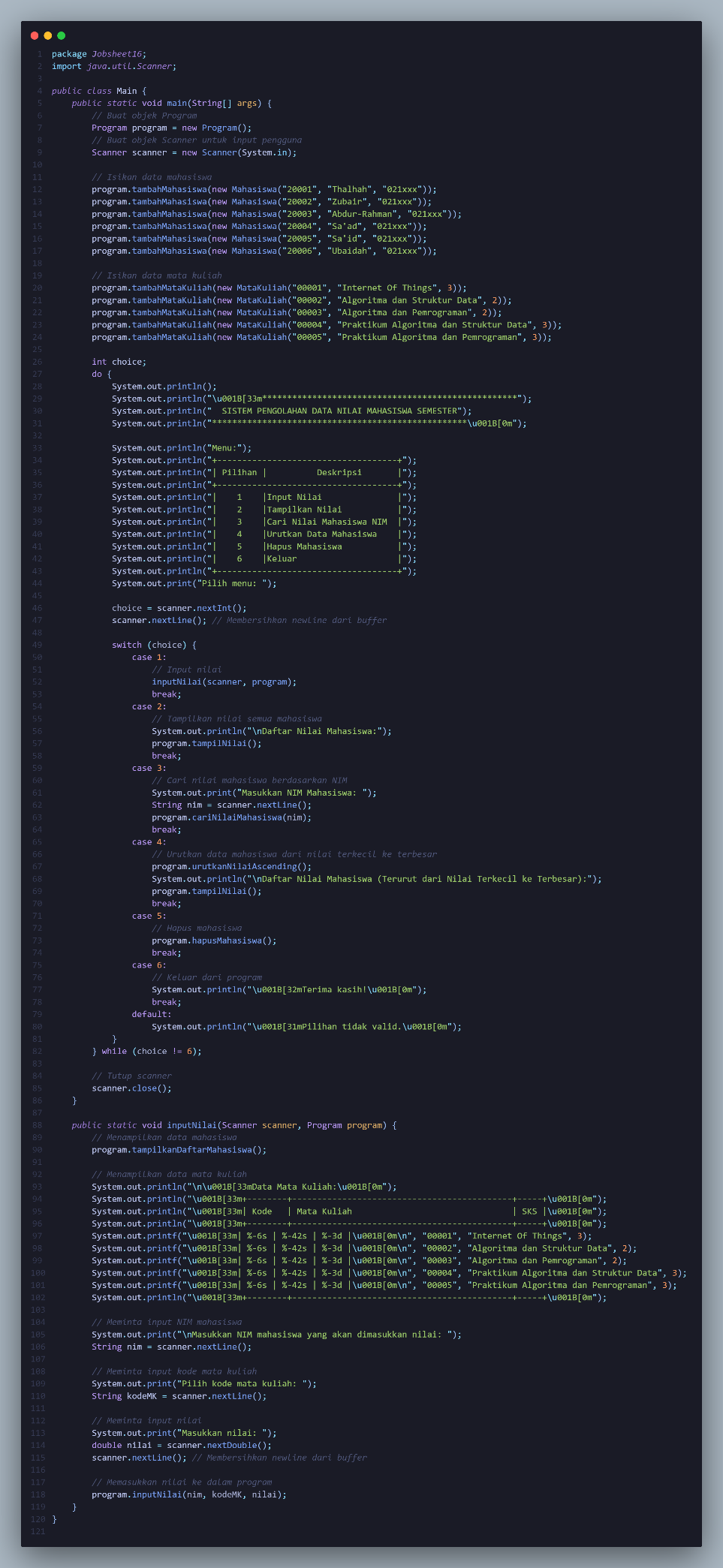
1. Untuk menambahkan prosedur penghapusan data mahasiswa menggunakan Queue pada collection di nomer 1, kita perlu melakukan beberapa langkah. Pertama, kita akan menggunakan **LinkedList** sebagai implementasi **Queue**. Kemudian, kita akan menambahkan method untuk menambahkan mahasiswa ke dalam **Queue** dan menghapus mahasiswa dari **Queue**. Berikut adalah modifikasi yang diperlukan:
2. Menggunakan LinkedList sebagai Queue
3. Menambahkan metode untuk menghapus data mahasiswa dari Queue
4. Menyesuaikan menu dan fungsi di dalam kelas Main untuk mengakomodasi penghapusan data mahasiswa.

Berikut adalah modifikasi nya:

* Class Program

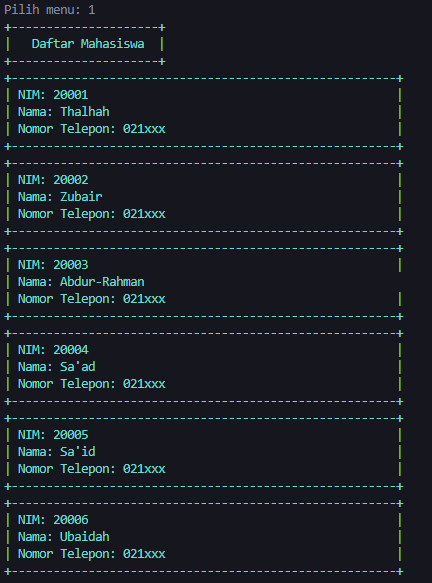


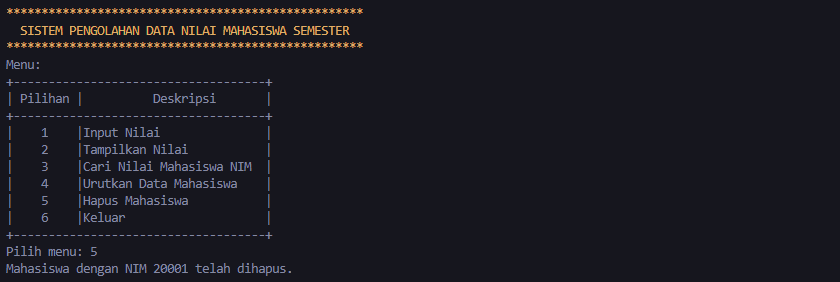
* Class Main

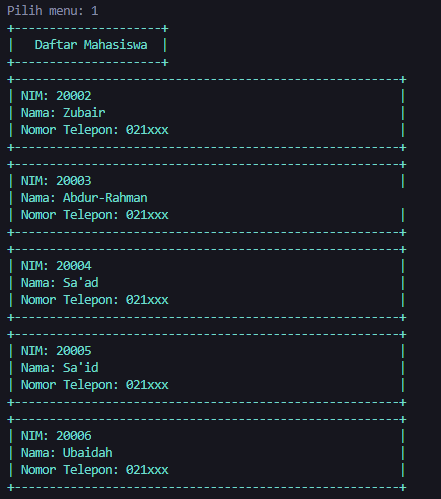


Kemudian berikut merupakan output dari program nya:

* **Sebelum di hapus**





* **Setelah penghapusan Data menggunakan Implementasi Queue**