**UJIAN AKHIR PRAKTIKUM**

**ALGORITMA STRUKTUR DATA**



Nama : Muhammad Azka Raki

NIM : 2311016110005

**KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI**

**UNIVERSITAS LAMBUNG MANGKURAT**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**PROGRAM STUDI S1 ILMU KOMPUTER**

**BANJARBARU**

**2024**

1. **Soal 1**

Buatlah program untuk mengelola data akademik mahasiswa dengan ketentuan berikut:

I. Struktur Data dan Format Data

A. Gunakan struktur data LinkedList untuk menyimpan data mahasiswa.

B. Setiap node dalam Linked List menyimpan informasi berikut:

● NIM (String)

● Nama (String)

● IPK (Double)

● Jumlah SKS yang Telah Ditempuh (Integer)

C. Buat daftar awal sebanyak 5 mahasiswa dengan data awal yang ditentukan pengguna. Pastikan setiap mahasiswa memiliki NIM yang unik.

II. Operasi Dasar yang Wajib Ada

A. Tambah Data Mahasiswa: Tambahkan node mahasiswa baru ke Linked List. Data yang dimasukkan adalah NIM, Nama, IPK, dan Jumlah SKS. Pastikan NIM tidak duplikat.

B. Hapus Data Mahasiswa Berdasarkan NIM: Cari mahasiswa berdasarkan NIM yang diinputkan pengguna, lalu hapus dari Linked List. Jika tidak ditemukan, tampilkan pesan error.

C. Cari Mahasiswa Berdasarkan Nama: Cari dan tampilkan semua mahasiswa yang namanya mengandung substring tertentu yang dimasukkan pengguna.

D. Tampilkan Rata-rata IPK: Hitung dan tampilkan rata-rata IPK dari semua mahasiswa dalam Linked List.

E. Urutkan Data Berdasarkan NIM: Urutkan Linked List berdasarkan NIM secara ascending.

III. Aturan Khusus Berdasarkan Digit NIM Gunakan digit Terakhir NIM untuk menentukan operasi tambahan berikut:

0. Tambahkan mahasiswa baru ke akhir Linked List. Data mahasiswa diinputkan oleh pengguna.

1. Hapus mahasiswa dengan NIM terkecil.

2. Cari mahasiswa dengan IPK tertinggi dan pindahkan ke awal Linked List.

3. Gandakan data mahasiswa terakhir di Linked List (salin data mahasiswa terakhir menjadi node baru).

4. Balikkan seluruh Linked List.

5. Hapus semua mahasiswa dengan IPK lebih kecil dari 2.0.

6. Jika jumlah mahasiswa ganjil, tambahkan mahasiswa dengan data berikut:

NIM: "9999999999", Nama: "Dummy", IPK: 4.0, SKS: 144. Jika jumlah mahasiswa genap, hapus mahasiswa dengan IPK terendah.

7. Cari mahasiswa dengan jumlah SKS terbanyak dan pindahkan ke posisi kedua dalam Linked List.

8. Hapus semua mahasiswa dengan NIM yang berisi angka ganjil di digit terakhir.

9. Gandakan data mahasiswa pertama di Linked List.

IV. Validasi Tambahan

● Jika pengguna mencoba menambahkan mahasiswa dengan NIM yang sudah ada, tampilkan pesan:

"Data mahasiswa dengan NIM ini sudah ada, silakan masukkan NIM yang berbeda."

● Jika pengguna mencoba menghapus atau mencari mahasiswa yang tidak ditemukan, tampilkan pesan:

"Data mahasiswa tidak ditemukan."

Source Code:

1. Main.java

|  |
| --- |
| package soal1;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  AkademikMahasiswa akademikMahasiswa = new AkademikMahasiswa();  System.out.println("\tProgram Akademik Mahasiswa");  String pilihan, nama, NIM;  do {  menu();  System.out.print("Masukkan pilihan = ");  pilihan = scan.nextLine();  switch (pilihan) {  case "1":  System.out.print("Masukkan Nama : ");  nama = scan.nextLine();  System.out.print("Masukkan NIM : ");  NIM = scan.nextLine();  System.out.print("Masukkan IPK : ");  double IPK = scan.nextDouble();  System.out.print("Masukkan SKS : ");  int sks = scan.nextInt();  akademikMahasiswa.tambahDataMahasiswa(NIM, nama, IPK, sks);  break;  case "2":  System.out.print("Masukkan NIM : ");  NIM = scan.nextLine();  scan.nextLine();  akademikMahasiswa.hapusDataMahasiswa(NIM);  break; case "3":  System.out.print("Masukkan Nama : ");  nama = scan.nextLine();  akademikMahasiswa.cariDataMahasiswa(nama);  break;  case "4":  akademikMahasiswa.tampilkanRataRataIPK();  break;  case "5":  akademikMahasiswa.urutkanBerdasarkanNIM();  break;  case "6":  akademikMahasiswa.hapusSemuaMahasiswaDenganIPKLebihKecilDari2();  break;  case "7":  return;  }  scan = new Scanner(System.in);  } while (true);  }  static void menu() {  System.out.println("1. Tambah Data Mahasiswa");  System.out.println("2. Hapus Data Mahasiswa");  System.out.println("3. Cari Mahasiswa Berdasarkan Nama");  System.out.println("4. Tampilkan Rata-rata IPK");  System.out.println("5. Urutkan Data Berdasarkan NIM");  System.out.println("6. Hapus Semua Mahasiswa Dengan IPK < 2");  System.out.println("7. Keluar");  }  } |

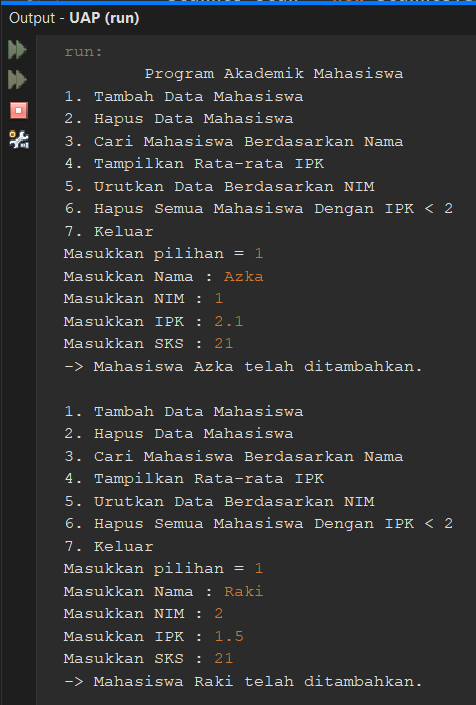
1. Mahasiswa.java

|  |
| --- |
| package soal1;  public class Mahasiswa {  private String NIM;  private String nama;  private double IPK;  private int sksYangTelahDitempuh;  public Mahasiswa(String NIM, String nama, double IPK, int sksYangTelahDitempuh) {  this.NIM = NIM;  this.nama = nama;  this.IPK = IPK;  this.sksYangTelahDitempuh = sksYangTelahDitempuh;  }  public String getNIM() {  return NIM;  }  public String getNama() {  return nama;  }  public double getIPK() {  return IPK;  }  public int getSksYangTelahDitempuh() {  return sksYangTelahDitempuh;  }        } |

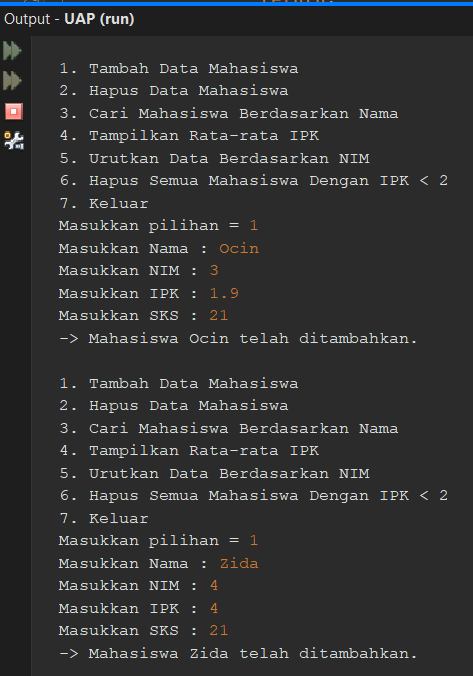
1. AkademikMahasiswa.java

|  |
| --- |
| package soal1;  import java.util.LinkedList;  public class AkademikMahasiswa {  LinkedList<Mahasiswa> listMahasiswa = new LinkedList<>();  void tambahDataMahasiswa(String NIM, String nama, double IPK, int sks) {  Mahasiswa mahasiswa = new Mahasiswa(NIM, nama, IPK, sks);  for (int i = 0; i < listMahasiswa.size(); i++) {  if (listMahasiswa.get(i).getNIM().equalsIgnoreCase(NIM)) {  System.out.println("Data mahasiswa dengan NIM ini sudah ada, silakan masukkan NIM yang berbeda.");  System.out.println("");  return;  }  }  listMahasiswa.add(mahasiswa);  System.out.println("-> Mahasiswa " + nama + " telah ditambahkan.");  System.out.println("");  }  void hapusDataMahasiswa(String NIM) {  for (int i = 0; i < listMahasiswa.size(); i++) {  if (listMahasiswa.get(i).getNIM().equalsIgnoreCase(NIM)) {  listMahasiswa.remove(i);  System.out.println("-> Data mahasiswa dengan NIM " + NIM + " telah dihapus");  System.out.println("");  return;  }  }  System.out.println("-> ERROR. Data mahasiswa tidak ditemukan.");  System.out.println("");  return;  }  void cariDataMahasiswa(String nama) {  boolean ketemu = false;  for (int i = 0; i < listMahasiswa.size(); i++) {  if (listMahasiswa.get(i).getNama().toLowerCase().contains(nama.toLowerCase())) {  Mahasiswa mahasiswa = listMahasiswa.get(i);  System.out.println(" Nama : " + mahasiswa.getNama());  System.out.println(" NIM : " + mahasiswa.getNIM());  System.out.println(" IPK : " + mahasiswa.getIPK());  System.out.println(" SKS : " + mahasiswa.getSksYangTelahDitempuh());  System.out.println("--------------------------------------------");  ketemu = true;  }  }  if (ketemu) {  return;  }  System.out.println("");  return;  }  void tampilkanRataRataIPK() {  double rataRata = 0;  for (int i = 0; i < listMahasiswa.size(); i++) {  Mahasiswa mahasiswa = listMahasiswa.get(i);  rataRata += mahasiswa.getIPK();  }  rataRata /= listMahasiswa.size();  System.out.println("Rata-rata IPK Mahasiswa = " + rataRata);  System.out.println("");  }  void urutkanBerdasarkanNIM() {  for (int i = 0; i < listMahasiswa.size() - 1; i++) {  for (int j = 0; j < listMahasiswa.size() - i - 1; j++) {  if (listMahasiswa.get(j).getNIM().compareToIgnoreCase(listMahasiswa.get(j + 1).getNIM()) > 0) {  Mahasiswa temp = listMahasiswa.get(j);  listMahasiswa.set(j, listMahasiswa.get(j + 1));  listMahasiswa.set(j + 1, temp);  }  }  }  System.out.println("-> Data mahasiswa telah diurutkan berdasarkan NIM secara ascending.");  System.out.println("");  }  void hapusSemuaMahasiswaDenganIPKLebihKecilDari2() {  boolean terhapus = false;  for (int i = 0; i < listMahasiswa.size(); i++) {  if (listMahasiswa.get(i).getIPK() < 2) {  listMahasiswa.remove(i);  i--;  terhapus = true;  }  }  if (terhapus) {  System.out.println("-> Semua mahasiswa dengan IPK kurang dari 2 telah dihapus.");  } else {  System.out.println("-> ERROR. Data mahasiswa tidak ditemukan.");  }  System.out.println("");  }  } |

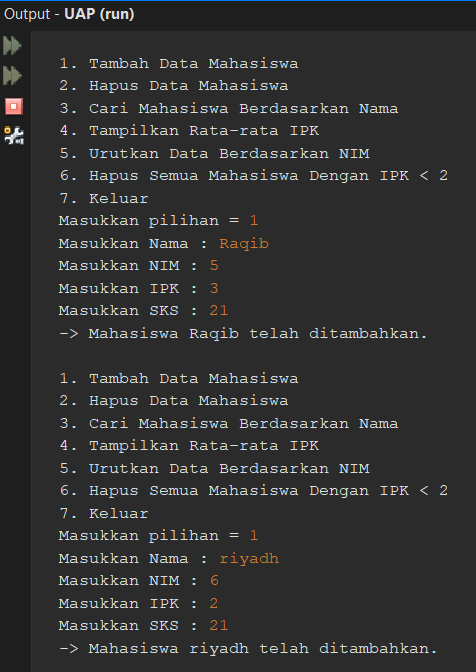
Output:



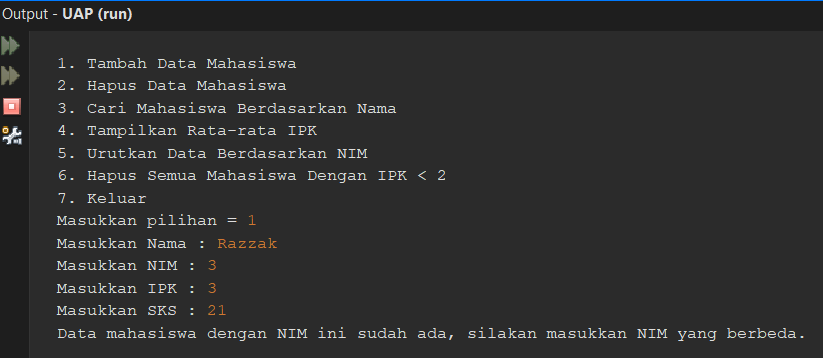
Gambar 1.1 hasil output



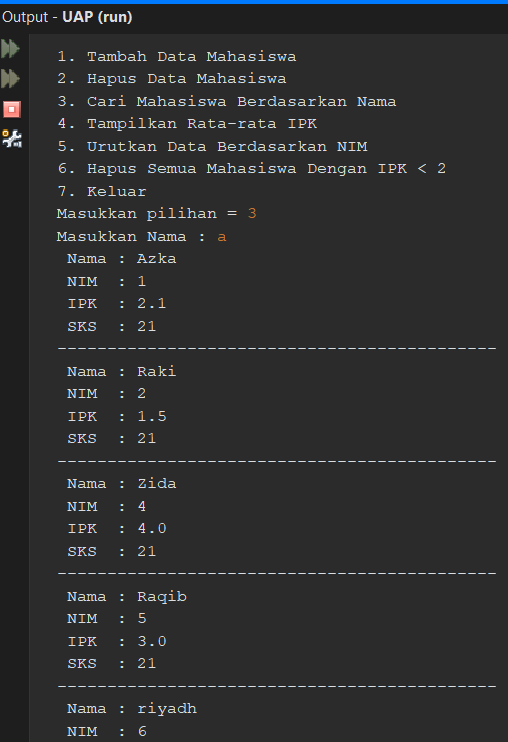
Gambar 1.2 hasil output



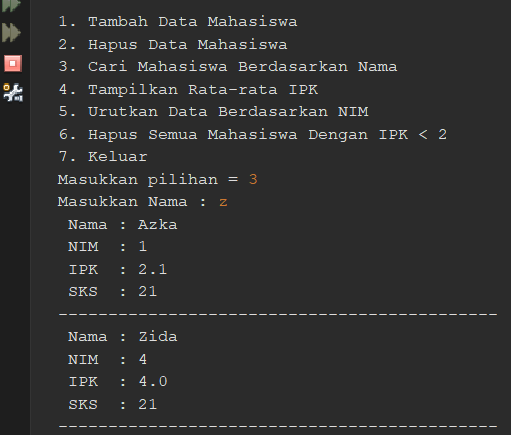
Gambar 1.3 hasil output



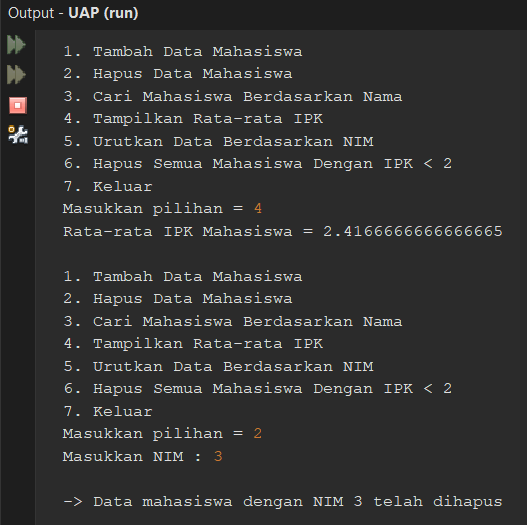
Gambar 1.4 hasil output



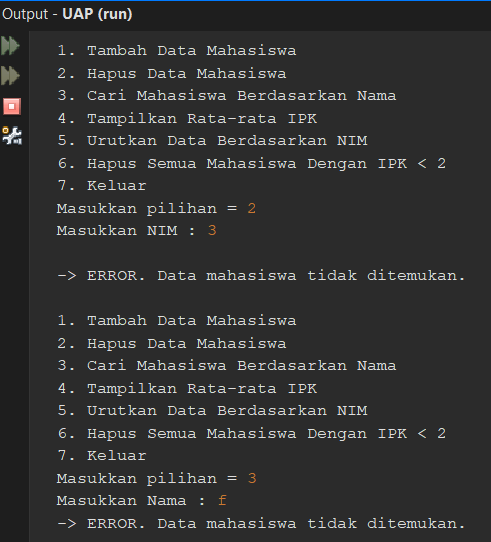
Gambar 1.5 hasil output



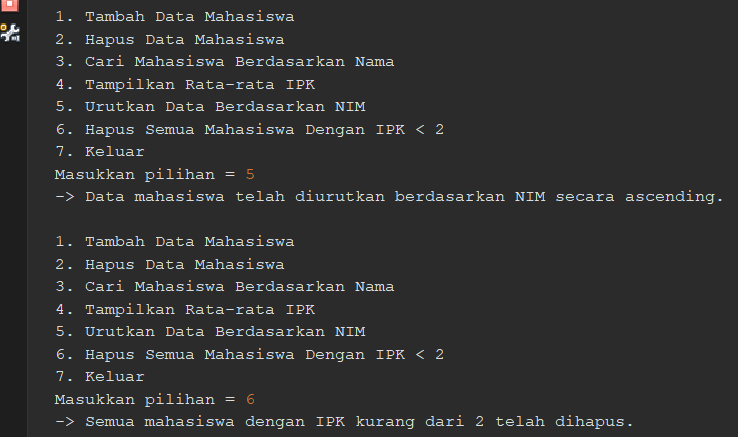
Gambar 1.6 hasil output



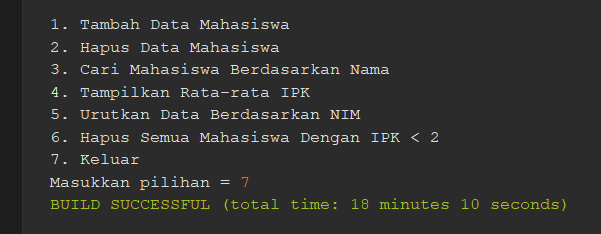
Gambar 1.7 hasil output



Gambar 1.8 hasil output



Gambar 1.9 hasil output



Gambar 1.10 hasil output

1. **Soal 2**

Buatlah program untuk mengelola persediaan barang di sebuah gudang menggunakan struktur data Linked List dengan ketentuan sebagai berikut:

I. Struktur Data dan Format Data

A. Gunakan struktur data Linked List untuk menyimpan informasi barang.

B. Setiap node dalam Linked List menyimpan informasi berikut:

● Kode Barang (String)

● Nama Barang (String)

● Harga Satuan (Double)

● Jumlah Stok (Integer)

II. Operasi Dasar yang Wajib Ada

A. Tambah Data Barang: Tambahkan node barang baru ke Linked List. Data yang dimasukkan adalah Kode Barang, Nama Barang, Jumlah Stok, dan Harga Satuan. Jika Kode Barang sudah ada, tampilkan pesan error.

B. Hapus Data Barang Berdasarkan Kode: Cari barang berdasarkan Kode Barang yang diinputkan pengguna, lalu hapus node tersebut dari Linked List. Jika tidak ditemukan, tampilkan pesan error.

C. Cari Barang Berdasarkan Nama: Cari barang yang namanya mengandung substring tertentu yang dimasukkan pengguna dan tampilkan datanya.

D. Tampilkan Total Nilai Stok Gudang: Hitung dan tampilkan total nilai stok semua barang (Jumlah Stok × Harga Satuan untuk setiap barang).

E. Urutkan Data Berdasarkan Kode Barang: Urutkan Linked List berdasarkan Kode Barang secara ascending.

III. Aturan Khusus Berdasarkan Digit NIM

Gunakan digit Terakhir NIM untuk menentukan operasi tambahan berikut:

0. Tambahkan barang baru dengan data berikut:

Kode Barang: "DEFAULT", Nama Barang: "Barang Default", Jumlah Stok: 50, Harga Satuan: 10,000.

1. Gandakan node barang pertama dalam Linked List.

2. Hapus barang dengan stok terkecil.

3. Balikkan urutan barang dalam Linked List.

4. Cari barang dengan harga satuan tertinggi dan pindahkan ke awal Linked List.

5. Tambahkan 20 ke stok barang terakhir dalam Linked List.

6. Jika jumlah barang dalam Linked List genap, tambahkan node baru dengan data: Kode Barang: "SPECIAL", Nama Barang: "Barang Khusus", Jumlah Stok: 100, Harga Satuan: 15,000. Jika ganjil, hapus barang dengan harga satuan terendah.

7. Hapus semua barang dengan jumlah stok lebih kecil dari 10.

8. Gandakan barang terakhir dalam Linked List.

9. Cari barang dengan harga satuan tertinggi, lalu gandakan node tersebut.

IV. Validasi Tambahan

● Jika pengguna mencoba menambahkan barang dengan Kode Barang yang sudah ada, tampilkan pesan:

"Barang dengan kode ini sudah ada. Silakan masukkan kode yang berbeda."

● Jika pengguna mencoba menghapus atau mencari barang yang tidak ditemukan, tampilkan pesan:

"Barang tidak ditemukan."

Source Code:

1. Main.java

|  |
| --- |
| package soal2;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  PersediaanBarang persediaanBarang = new PersediaanBarang();  double hargaSatuan;  int jumlahStok;  String pilihan, kode, nama;  do {  menu();  System.out.print("Masukkan pilihan : ");  pilihan = scan.nextLine();  switch (pilihan) {  case "1":  System.out.print("Masukkan Kode : ");  kode = scan.nextLine();  System.out.print("Masukkan Nama : ");  nama = scan.nextLine();  System.out.print("Masukkan Harga Satuan : ");  hargaSatuan = scan.nextDouble();  System.out.print("Masukkan Jumlah Stok : ");  jumlahStok = scan.nextInt();  persediaanBarang.tambahBarang(kode, nama, hargaSatuan, jumlahStok);  break;  case "2":  System.out.print("Masukkan Kode : ");  kode = scan.nextLine();  persediaanBarang.hapusBarang(kode);  break;  case "3":  System.out.print("Masukkan Nama : ");  nama = scan.nextLine();  persediaanBarang.cariBarangBerdasarkanNama(nama);  break;  case "4":  persediaanBarang.tampilkanTotalNilaiStokGudang();  break;  case "5":  persediaanBarang.urutkanBerdasarkanKode();  break;  case "6":  persediaanBarang.tambah20StokKeBarangTerakhir();  break;  case "7":  return;  }  scan = new Scanner(System.in);  } while (true);  }  static void menu() {  System.out.println("1. Tambah Data Barang");  System.out.println("2. Hapus Data Barang");  System.out.println("3. Cari Barang Berdasarkan Nama");  System.out.println("4. Tampilkan Total Nilai Stok Gudang");  System.out.println("5. Urutkan Data Berdasarkan Kode Barang");  System.out.println("6. Tambahkan 20 ke Stok Barang Terakhir");  System.out.println("7. Keluar");  }  } |

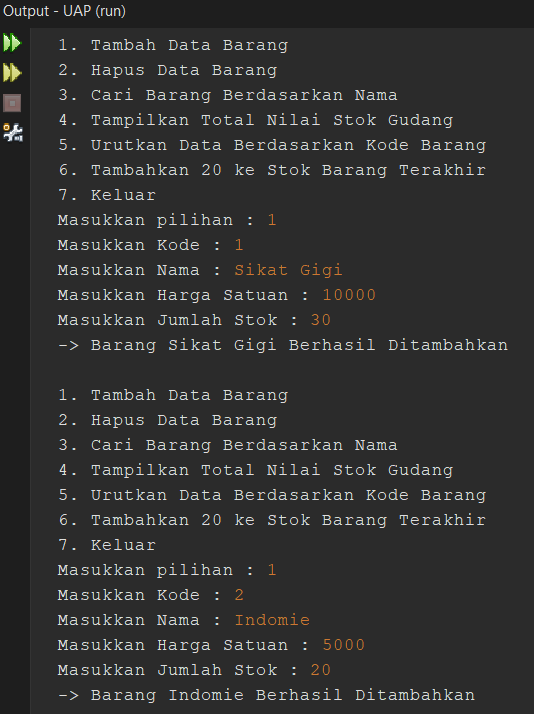
1. Barang.java

|  |
| --- |
| package soal2;  public class Barang {  private String kode;  private String nama;  private double hargaSatuan;  private int jumlahStok;  public Barang(String kode, String nama, double hargaSatuan, int jumlahStok) {  this.kode = kode;  this.nama = nama;  this.hargaSatuan = hargaSatuan;  this.jumlahStok = jumlahStok;  }  public String getKode() {  return kode;  }  public String getNama() {  return nama;  }  public double getHargaSatuan() {  return hargaSatuan;  }  public int getJumlahStok() {  return jumlahStok;  }  public void setJumlahStok(int jumlahStok) {  this.jumlahStok = jumlahStok;  }  } |

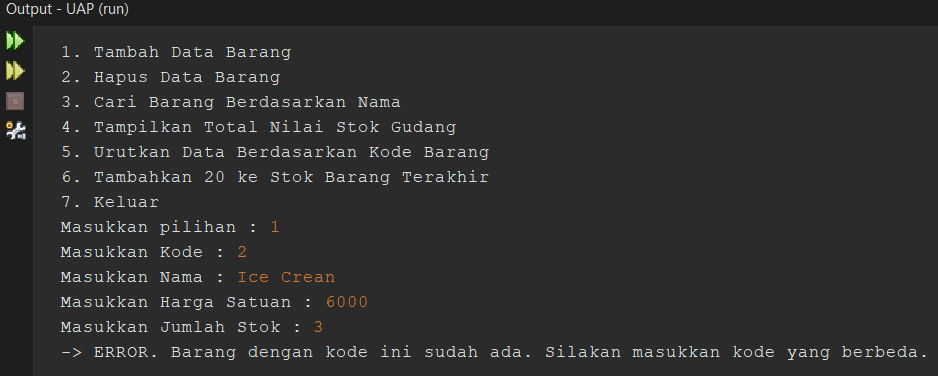
1. PersediaanBarang.java

|  |
| --- |
| package soal2;  import java.util.LinkedList;  public class PersediaanBarang {  private LinkedList<Barang> listBarang = new LinkedList<>();  void tambahBarang(String kode, String nama, double hargaSatuan, int jumlahStok) {  for (int i = 0; i < listBarang.size(); i++) {  Barang barang = listBarang.get(i);  if (barang.getKode().toLowerCase().contains(kode.toLowerCase())) {  System.out.println("-> ERROR. Barang dengan kode ini sudah ada. Silakan masukkan kode yang berbeda.");  System.out.println("");  return;  }  }  Barang barang = new Barang(kode, nama, hargaSatuan, jumlahStok);  listBarang.add(barang);  System.out.println("-> Barang " + nama + " Berhasil Ditambahkan");  System.out.println("");  }  void hapusBarang(String kode) {  for (int i = 0; i < listBarang.size(); i++) {  Barang barang = listBarang.get(i);  if (barang.getKode().equals(kode)) {  listBarang.remove(i);  System.out.println("-> Barang dengan kode " + kode + " Berhasil Dihapus");  System.out.println("");  return;  }  }  System.out.println("-> ERROR. Barang tidak ditemukan.");  System.out.println("");  }  void cariBarangBerdasarkanNama(String nama) {  boolean ketemu = false;  for (int i = 0; i < listBarang.size(); i++) {  if (listBarang.get(i).getNama().toLowerCase().contains(nama.toLowerCase())) {  Barang barang = listBarang.get(i);  System.out.println(" Kode : " + barang.getKode());  System.out.println(" Nama : " + barang.getNama());  System.out.println(" Harga Satuan : " + barang.getHargaSatuan());  System.out.println(" Jumlah Stok : " + barang.getJumlahStok());  System.out.println("--------------------------------------------");  ketemu = true;  }  }  if (ketemu) {  return;  }  System.out.println("-> ERROR. Data mahasiswa tidak ditemukan.");  System.out.println("");  }  void tampilkanTotalNilaiStokGudang() {  double total = 0;  for (int i = 0; i < listBarang.size(); i++) {  Barang barang = listBarang.get(i);  total += barang.getJumlahStok() \* barang.getHargaSatuan();  }  System.out.println("Total Nilai Stok Gudang : Rp " + total);  System.out.println("");  }  void urutkanBerdasarkanKode() {  for (int i = 0; i < listBarang.size() - 1; i++) {  for (int j = 0; j < listBarang.size() - i - 1; j++) {  if (listBarang.get(j).getKode().compareToIgnoreCase(listBarang.get(j + 1).getKode()) > 0) {  Barang temp = listBarang.get(j);  listBarang.set(j, listBarang.get(j + 1));  listBarang.set(j + 1, temp);  }  }  }  System.out.println("-> Data barang telah diurutkan berdasarkan kode secara ascending.");  System.out.println("");  }  void tambah20StokKeBarangTerakhir() {  Barang barang = listBarang.getLast();  barang.setJumlahStok(barang.getJumlahStok() + 20);  listBarang.set(listBarang.size() - 1, barang);  System.out.println("-> Stok Barang Terakhir Telah Ditambah 20.");  System.out.println("");  }  } |

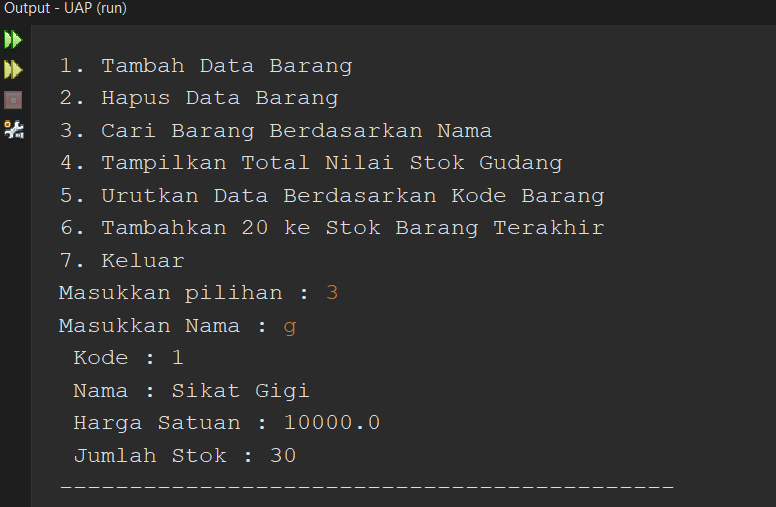
Output:



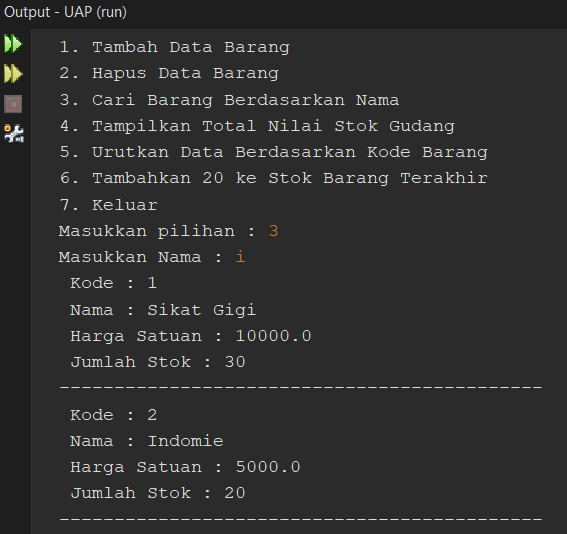
Gambar 2.1 hasil output

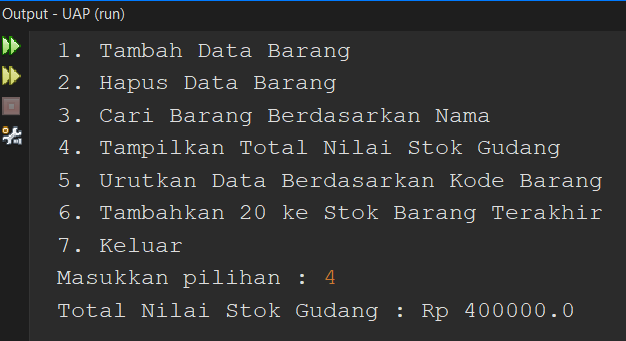


Gambar 2.2 hasil output

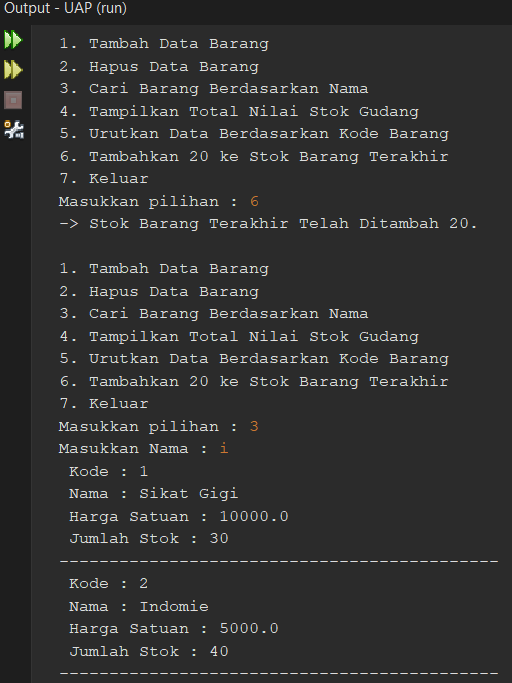


Gambar 2.3 hasil output

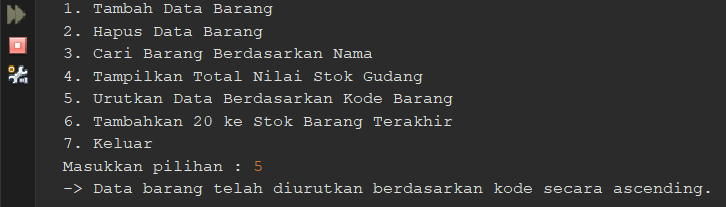
  
Gambar 2.4 hasil output



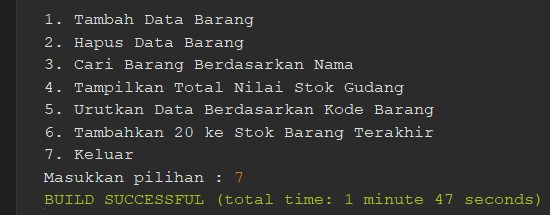
Gambar 2.5 hasil output



Gambar 2.6 hasil output



Gambar 2.6 hasil output



Gambar 2.7 hasil output

1. **Soal 3**

Buatlah program untuk mengelola antrian pasien di rumah sakit dengan requirement berikut:

a. Menggunakan struktur data “Queue”

b. Gunakan NIM untuk menentukan aturan khusus yang dapat mengubah urutan pasien dalam Queue

c. Buat daftar ID pasien awal sebanyak 5 berupa bilangan bulat.

d. Setiap digit NIM menentukan tindakan pada queue sebagai berikut:

0 : Keluarkan pasien pertama dari antrian

1 : Tambahkan pasien dengan ID baru menggunakan input user

2 : Tukar posisi dua pasien pertama.

3 : Gandakan ID pasien terakhir

4 : Pindahkan pasien terakhir ke awal antrian

5 : keluarkan pasien dengan ID terkecil dari antrian

6 : Jika jumlah pasien genap, keluarkan pasien tengah, jika ganjil maka

tambahkan ID baru 999 ke antrian

7 : Balikkan seluruh antrian

8 : Hapus semua pasien dengan ID lebih kecil dari 10

9 : Jika ada pasien dengan ID lebih besar dari 999, hapus pasien tersebut

dari antrian

Source Code:

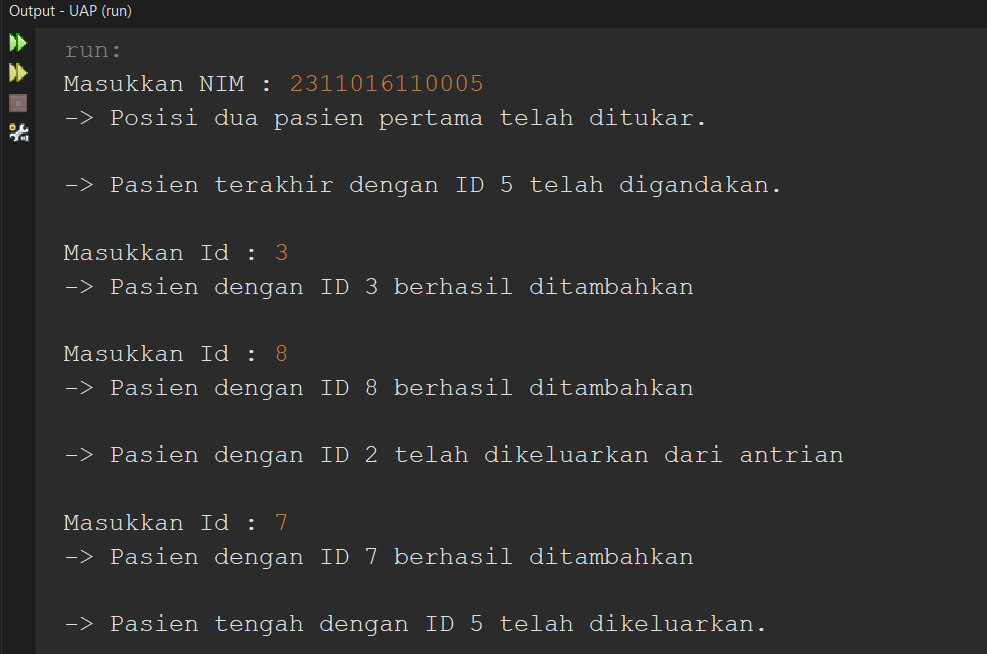
1. Main.java

|  |
| --- |
| package soal3;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  AntrianPasien antrianPasien = new AntrianPasien();  System.out.print("Masukkan NIM : ");  String[] arrStrNIM = scan.nextLine().split("");  for (int i = 0; i < arrStrNIM.length; i++) {  int digitNIM = Integer.parseInt(arrStrNIM[i]);  switch (digitNIM) {  case 0:  antrianPasien.keluarkanPasienPertama();  break;  case 1:  System.out.print("Masukkan Id : ");  int id = scan.nextInt();  antrianPasien.tambahPasien(id);  break;  case 2:  antrianPasien.tukarPosisiDuaPasienPertama();  break;  case 3:  antrianPasien.gandakanIdPasienTerakhir();  break;  case 4:  antrianPasien.pindahPasienTerakhirKeAwal();  break;  case 5:  antrianPasien.keluarkanPasienDenganIdTerkecil();  break;  case 6:  antrianPasien.tindakanKhususNomor6();  break;  case 7:  antrianPasien.balik();  break;  case 8:  antrianPasien.hapusSemuaPasienDenganIDLebihKecilDari10();  break;  case 9:  antrianPasien.tindakanKhususNomor9();  break;  }  }  }  } |

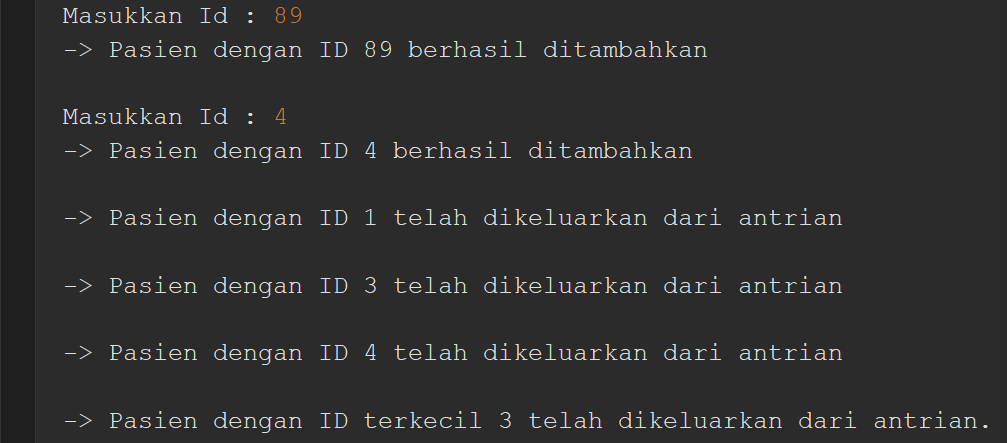
1. AntrianPasien.java

|  |
| --- |
| package soal3;  import java.util.Queue;  import java.util.LinkedList;  public class AntrianPasien {  private Queue<Integer> antrianPasien = new LinkedList<>();  public AntrianPasien() {  for (int i = 1; i <= 5; i++) {  antrianPasien.add(i);  }  }  void keluarkanPasienPertama() {  int IdPasien = antrianPasien.remove();  System.out.println("-> Pasien dengan ID " + IdPasien + " telah dikeluarkan dari antrian");  System.out.println("");  }  void tambahPasien(int id) {  antrianPasien.add(id);  System.out.println("-> Pasien dengan ID " + id + " berhasil ditambahkan");  System.out.println("");  }  void tukarPosisiDuaPasienPertama() {  int pasienPertama = antrianPasien.remove();  int pasienKedua = antrianPasien.remove();  antrianPasien.add(pasienKedua);  antrianPasien.add(pasienPertama);  for (int i = 0; i < antrianPasien.size() - 2; i++) {  antrianPasien.add(antrianPasien.remove());  }  System.out.println("-> Posisi dua pasien pertama telah ditukar.");  System.out.println("");  }  void gandakanIdPasienTerakhir() {  int pasienTerakhir = 0;  for (int id : antrianPasien) {  pasienTerakhir = id;  }  antrianPasien.add(pasienTerakhir);  System.out.println("-> Pasien terakhir dengan ID " + pasienTerakhir + " telah digandakan.");  System.out.println("");  }  void pindahPasienTerakhirKeAwal() {  Queue<Integer> antrianPasienSementara = new LinkedList<>();  int pasienTerakhir = 0;  for (int id : antrianPasien) {  pasienTerakhir = id;  }  while (!antrianPasien.isEmpty()) {  int id = antrianPasien.remove();  if (id != pasienTerakhir) {  antrianPasienSementara.add(id);  }  }  antrianPasien.add(pasienTerakhir);  while (!antrianPasienSementara.isEmpty()) {  antrianPasien.add(antrianPasienSementara.remove());  }  System.out.println("-> Pasien terakhir dengan ID " + pasienTerakhir + " telah dipindahkan ke awal antrian.");  System.out.println("");  }  void keluarkanPasienDenganIdTerkecil() {  int minId = Integer.MAX\_VALUE;  Queue<Integer> antrianSementara = new LinkedList<>();  boolean idTerkecilTerhapus = false;  for (int id : antrianPasien) {  if (id < minId) {  minId = id;  }  }  while (!antrianPasien.isEmpty()) {  int id = antrianPasien.remove();  if (id == minId && !idTerkecilTerhapus) {  idTerkecilTerhapus = true;  } else {  antrianSementara.add(id);  }  }  antrianPasien = antrianSementara;  System.out.println("-> Pasien dengan ID terkecil " + minId + " telah dikeluarkan dari antrian.");  System.out.println("");  }  void tindakanKhususNomor6() {  Queue<Integer> antrianSementara = new LinkedList<>();  int jumlahPasien = antrianPasien.size();  if (jumlahPasien % 2 == 0) {  int indeksTengah = (jumlahPasien / 2) - 1;  for (int i = 0; i < jumlahPasien; i++) {  int id = antrianPasien.remove();  if (i == indeksTengah) {  System.out.println("-> Pasien tengah dengan ID " + id + " telah dikeluarkan.");  } else {  antrianSementara.add(id);  }  }  antrianPasien = antrianSementara;  } else {  antrianPasien.add(999);  System.out.println("-> Jumlah pasien ganjil, ID baru 999 telah ditambahkan ke antrian.");  }  System.out.println("");  }  void balik() {  int n = antrianPasien.size();  int[] arrayPasien = new int[n];  int index = 0;  while (!antrianPasien.isEmpty()) {  arrayPasien[index++] = antrianPasien.remove();  }  for (int i = n - 1; i >= 0; i--) {  antrianPasien.add(arrayPasien[i]);  }  System.out.println("-> Antrian telah dibalik.");  System.out.println("");  }  void hapusSemuaPasienDenganIDLebihKecilDari10() {  Queue<Integer> antrianPasienSementara = new LinkedList<>();  while (!antrianPasien.isEmpty()) {  int id = antrianPasien.remove();  if (id >= 10) {  antrianPasienSementara.add(id);  }  }  antrianPasien = antrianPasienSementara;  System.out.println("-> Semua pasien dengan ID lebih kecil dari 10 telah dihapus dari antrian.");  System.out.println("");  }  void tindakanKhususNomor9() {  Queue<Integer> antrianPasienSementara = new LinkedList<>();  while (!antrianPasien.isEmpty()) {  int id = antrianPasien.remove();  if (id <= 999) {  antrianPasienSementara.add(id);  }  }  antrianPasien = antrianPasienSementara;  System.out.println("-> Semua pasien dengan ID lebih besar dari 999 telah dihapus dari antrian.");  System.out.println("");  }  } |

Output:



Gambar 3.1 hasil output



Gambar 3.2 hasil output

1. **Soal 4**

Buatkan program untuk manajemen produksi pabrik dengan requirement berikut:

a. Menggunakan struktur data “Queue”

b. Buat daftar baku awal sebanyak 5 yang diwakili oleh nomor seri bahan baku berupa bilangan bulat.

c. Gunakan input user untuk melakukan pemrosesan.

d. Aturan pemrosesan bahan baku adalah sebagai berikut:

- Tambah (A 60): Tambahkan bahan baku dengan nomor seri 60 ke akhir antrian

- Keluarkan (R): Keluarkan bahan baku pertama dari antrian

- Duplikasi (D): Gandakan bahan baku pertama dalam antrian dan tambahkan ke akhir

- Gabungkan (M): Ambil dua bahan baku pertama, gabungkan nomor serinya dengan menjumlahkan, lalu masukkan hasilnya kembali ke antrian

- Pindahkan ke Awal (F 60): Jika bahan baku dengan nomor seri 60 ada di antrian, pindahkan ke posisi paling depan.

- Balikkan (B): Balikkan seluruh antrian

- Hapus Ganjil (O): Hapus semua bahan baku dengan nomor seri ganjil dari antrian

- Hapus Genap (E): Hapus semua bahan baku dengan nomor seri genap dari antrian

Source Code:

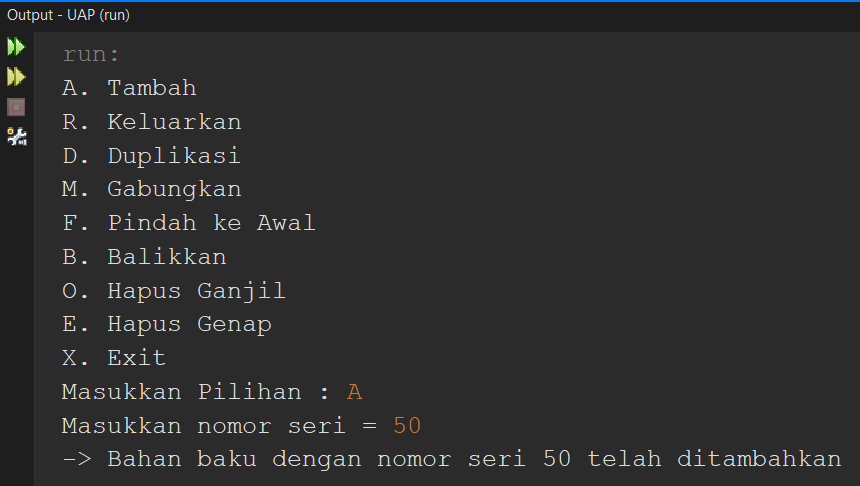
1. Main.java

|  |
| --- |
| package soal4;  import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  Pabrik pabrik = new Pabrik();  String pilihan;  int nomorSeri;  do {  menu();  System.out.print("Masukkan Pilihan : ");  pilihan = scan.nextLine();  switch (pilihan) {  case "A":  System.out.print("Masukkan nomor seri = ");  nomorSeri = scan.nextInt();  pabrik.tambah(nomorSeri);  break;  case "R":  pabrik.keluarkan();  break;  case "D":  pabrik.duplikasi();  break;  case "M":  pabrik.gabung();  break;  case "F":  System.out.print("Masukkan nomor seri = ");  nomorSeri = scan.nextInt();  pabrik.pindahKeAwal(nomorSeri);  break;  case "B":  pabrik.balik();  break;  case "O":  pabrik.hapusNomorSeriGanjil();  break;  case "E":  pabrik.hapusNomorSeriGenap();  break;  case "X":  return;  default:  System.out.println("");  break;  }  scan = new Scanner(System.in);  } while (true);  }  static void menu() {  System.out.println("A. Tambah");  System.out.println("R. Keluarkan");  System.out.println("D. Duplikasi");  System.out.println("M. Gabungkan");  System.out.println("F. Pindah ke Awal");  System.out.println("B. Balikkan");  System.out.println("O. Hapus Ganjil");  System.out.println("E. Hapus Genap");  System.out.println("X. Exit");  }  } |

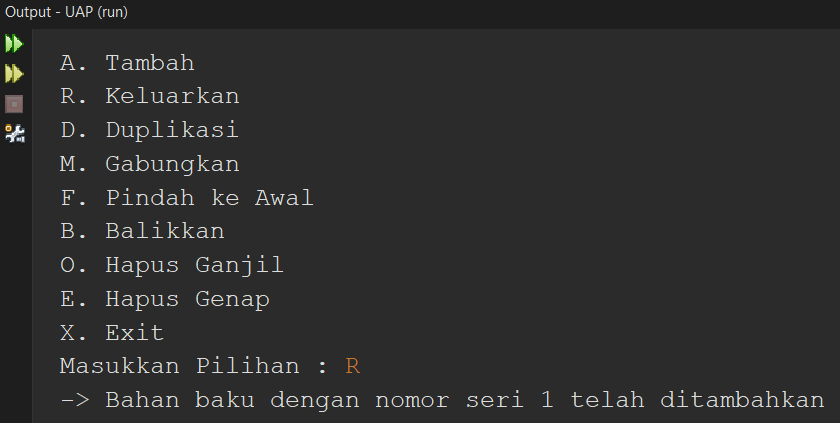
1. Pabrik.java

|  |
| --- |
| package soal4;  import java.util.Queue;  import java.util.LinkedList;  public class Pabrik {  private Queue<Integer> pabrik = new LinkedList<>();  public Pabrik() {  for (int i = 1; i <= 5; i++) {  pabrik.add(i);  }  }  void tambah(int nomorSeri) {  pabrik.add(nomorSeri);  System.out.println("-> Bahan baku dengan nomor seri " + nomorSeri + " telah ditambahkan");  System.out.println("");  }  void keluarkan() {  int nomorSeri = pabrik.remove();  System.out.println("-> Bahan baku dengan nomor seri " + nomorSeri + " telah ditambahkan");  System.out.println("");  }  void duplikasi() {  int nomorSeri = pabrik.peek();  pabrik.add(nomorSeri);  System.out.println("-> Bahan baku dengan nomor seri " + nomorSeri + " telah diduplikat");  System.out.println("");  }  void gabung() {  int nomorSeriBahanPertama = pabrik.remove();  int nomorSeriBahanKedua = pabrik.remove();  int hasilGabung = nomorSeriBahanPertama + nomorSeriBahanKedua;  pabrik.add(hasilGabung);  pabrik.add(nomorSeriBahanPertama);  pabrik.add(nomorSeriBahanKedua);  System.out.println("-> Bahan baku dengan nomor seri " + nomorSeriBahanPertama + " dan " + nomorSeriBahanKedua  + "\ntelah digabung menjadi " + hasilGabung);  System.out.println("");  }  void pindahKeAwal(int nomorSeri) {  Queue<Integer> antrianSementara = new LinkedList<>();  boolean ketemu = false;  while (!pabrik.isEmpty()) {  int nomorSeriBahan = pabrik.remove();  if (nomorSeriBahan == nomorSeri && !ketemu) {  ketemu = true;  } else {  antrianSementara.add(nomorSeriBahan);  }  }  if (ketemu) {  pabrik.add(nomorSeri);  isiPabrik(antrianSementara);  System.out.println("-> Bahan baku dengan nomor seri " + nomorSeri + " telah dipindahkan ke posisi paling depan.");  } else {  isiPabrik(antrianSementara);  System.out.println("-> Bahan baku dengan nomor seri " + nomorSeri + " tidak ditemukan.");  }  System.out.println("");  }  void balik() {  int[] arraySementara = new int[pabrik.size()];  for (int i = 0; i < pabrik.size(); i++) {  int nomorSeri = pabrik.poll();  arraySementara[i] = nomorSeri;  }  for (int i = arraySementara.length - 1; i >= 0; i--) {  pabrik.add(arraySementara[i]);  }  System.out.println("-> Antrian telah dibalikkan");  System.out.println("");  }  void hapusNomorSeriGanjil() {  Queue<Integer> antrianSementara = new LinkedList<>();  boolean ketemu = false;  while (!pabrik.isEmpty()) {  int nomorSeriBahan = pabrik.remove();  if (nomorSeriBahan % 2 == 1 && !ketemu) {  ketemu = true;  } else {  antrianSementara.add(nomorSeriBahan);  }  }  isiPabrik(antrianSementara);  System.out.println("-> Semua Bahan Baku Dengan Nomor Seri Ganjil Telah Terhapus");  System.out.println("");  }  void hapusNomorSeriGenap() {  Queue<Integer> antrianSementara = new LinkedList<>();  boolean ketemu = false;  while (!pabrik.isEmpty()) {  int nomorSeriBahan = pabrik.remove();  if (nomorSeriBahan % 2 == 0 && !ketemu) {  ketemu = true;  } else {  antrianSementara.add(nomorSeriBahan);  }  }  isiPabrik(antrianSementara);  System.out.println("-> Semua Bahan Baku Dengan Nomor Seri Genap Telah Terhapus");  System.out.println("");  }  private void isiPabrik(Queue<Integer> antrian) {  for (int nomorSeri : antrian) {  pabrik.add(nomorSeri);  }  }  } |

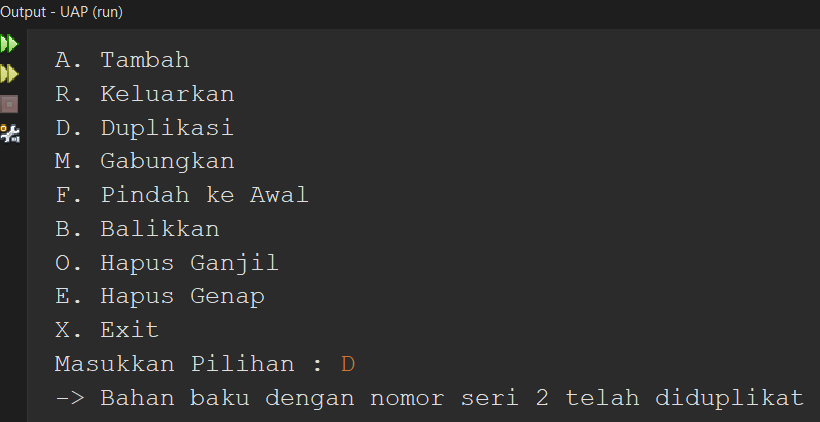
Output:



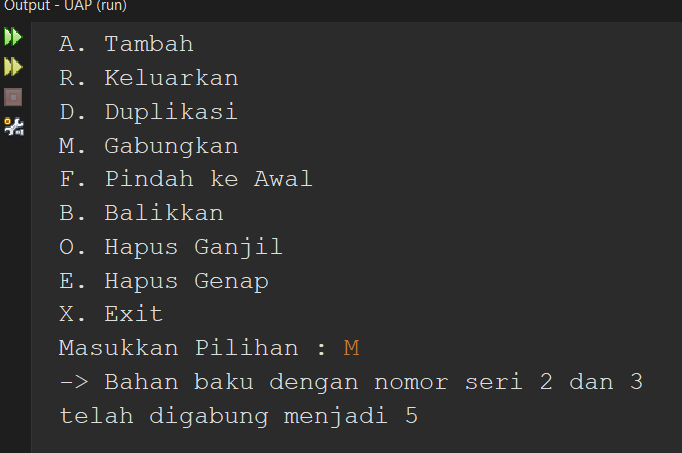
Gambar 4.1 hasil output



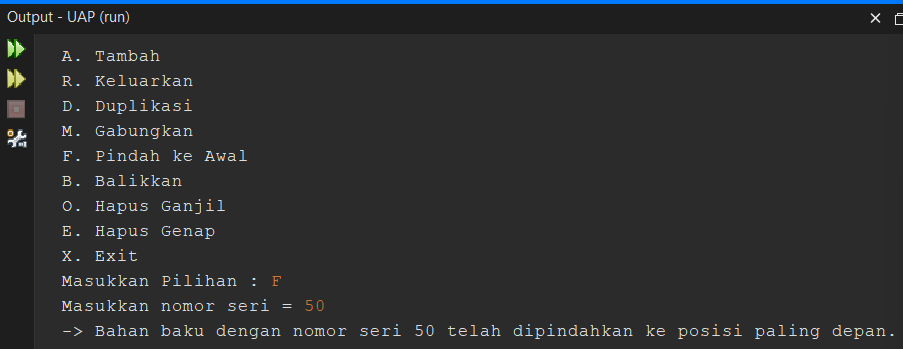
Gambar 4.2 hasil output



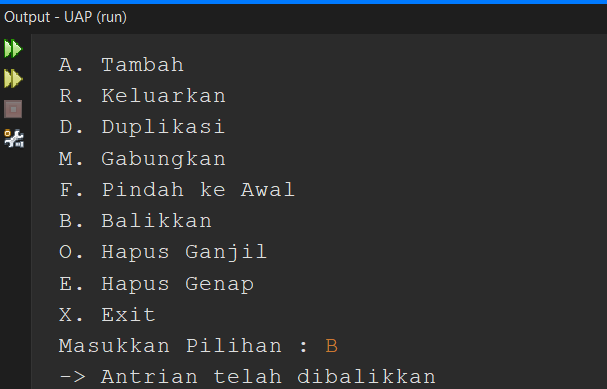
Gambar 4.3 hasil output



Gambar 4.4 hasil output



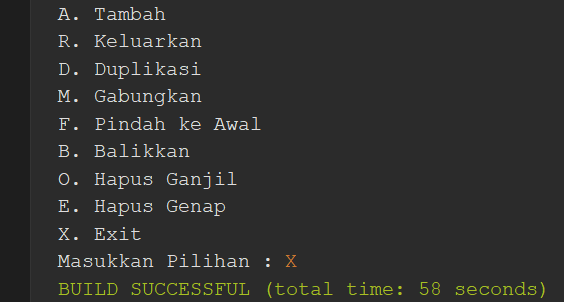
Gambar 4.5 hasil output



Gambar 4.6 hasil output



Gambar 4.7 hasil output



Gambar 4.8 hasil output

1. **Soal 5**

Anda adalah seorang Robotic Engineer di sebuah perusahaan manufacturing yang bertugas untuk mengotomasi proses supply and chain produk menggunakan robot cerdas. Untuk mengimplementasikan permasalahan tersebut, pertama-tama Anda harus mengetahui bahwa Anda akan menyortir kode produk berdasarkan nomor ganjil atau genap menggunakan struktur data Stack atau secara bertumpuk. Berikut adalah requirement yang diberikan:

- kode\_produk = [(NIM Anda)(Tanggal Lahir & Tahun Lahir Anda)].

Misal: [2,3,1,1,0,1,6,3,1,0,0,0,3,3,1,2,0,0,4]

- Jika kode produk adalah nomor ganjil, maka robot akan memasukkan ke stack bernama stack\_ganjil, dan sebaliknya bernama stack\_genap. Jika terdapat kode produk dengan nomor 0, maka masukkan ke stack\_tidak\_diproses.

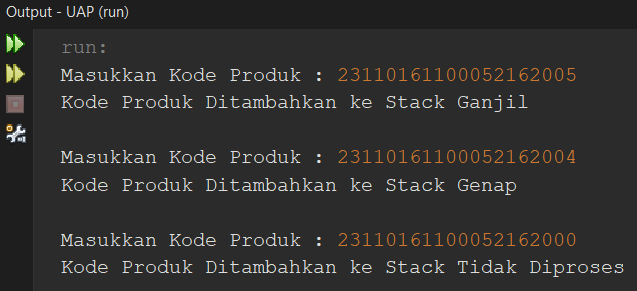
- Stack yang digunakan adalah stack tidak terbatas.

Source Code:

1. Main.java

|  |
| --- |
| package soal5;  import java.util.Scanner;  import java.util.Stack;  public class Main {  static Stack<Integer[]> stackGenap = new Stack<>();  static Stack<Integer[]> stackGanjil = new Stack<>();  static Stack<Integer[]> stackTidakDiproses = new Stack<>();  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  do {  System.out.print("Masukkan Kode Produk : ");  String[] kode = scan.nextLine().split("");  Integer[] arrKode = new Integer[kode.length];  for (int i = 0; i < kode.length; i++) {  arrKode[i] = Integer.valueOf(kode[i]);  }  sort(arrKode);  } while (true);  }  static void sort(Integer[] arr) {  if (arr[arr.length - 1] % 2 == 0 && arr[arr.length - 1] != 0) {  stackGenap.add(arr);  System.out.println("Kode Produk Ditambahkan ke Stack Genap");  } else if (arr[arr.length - 1] % 2 == 1) {  stackGanjil.add(arr);  System.out.println("Kode Produk Ditambahkan ke Stack Ganjil");  } else {  stackTidakDiproses.add(arr);  System.out.println("Kode Produk Ditambahkan ke Stack Tidak Diproses");  }  System.out.println("");  }  } |

Output:



Gambar 5.1 hasil output

1. **Soal 6**

Sebuah gudang besar memiliki beberapa unit penyimpanan, yang masing-masing direpresentasikan dengan struktur data stack. Setiap unit penyimpanan memiliki kapasitas maksimum [2 digit terakhir NIM] barang. Gudang besar memiliki tiga unit penyimpanan, yaitu unit penyimpanan utama, unit penyimpanan cadangan, dan unit penyimpanan yang tidak dapat diterima (rejected). Jika unit penyimpanan utama penuh, maka akan dialihkan ke unit penyimpanan cadangan, jika penuh juga, maka alihkan ke unit penyimpanan yang tidak dapat diterima. Buatlah simulasi gudang besar tersebut menggunakan struktur data Stack dengan ketentuan:

a. Barang memiliki atribut kode produk, Anda bebas menentukannya. Misalnya: A0013, A002, A109

b. Unit penyimpanan tidak boleh melebihi kapasitas karena akan menyebabkan stackoverflow

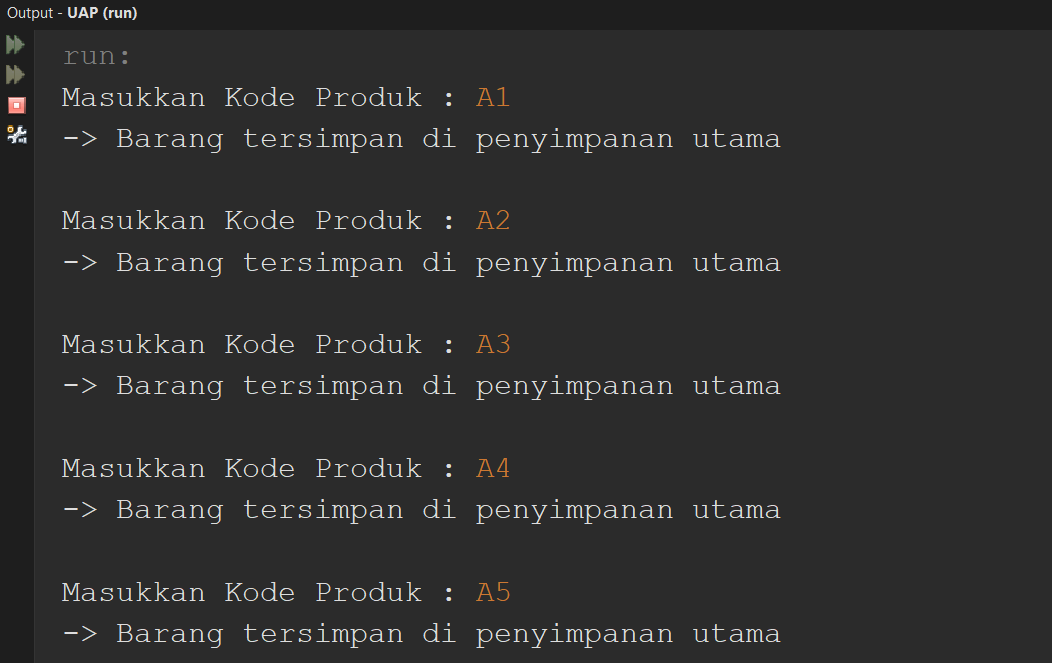
c. Melakukan simulasi hingga ada barang yang di-rejected

Source Code:

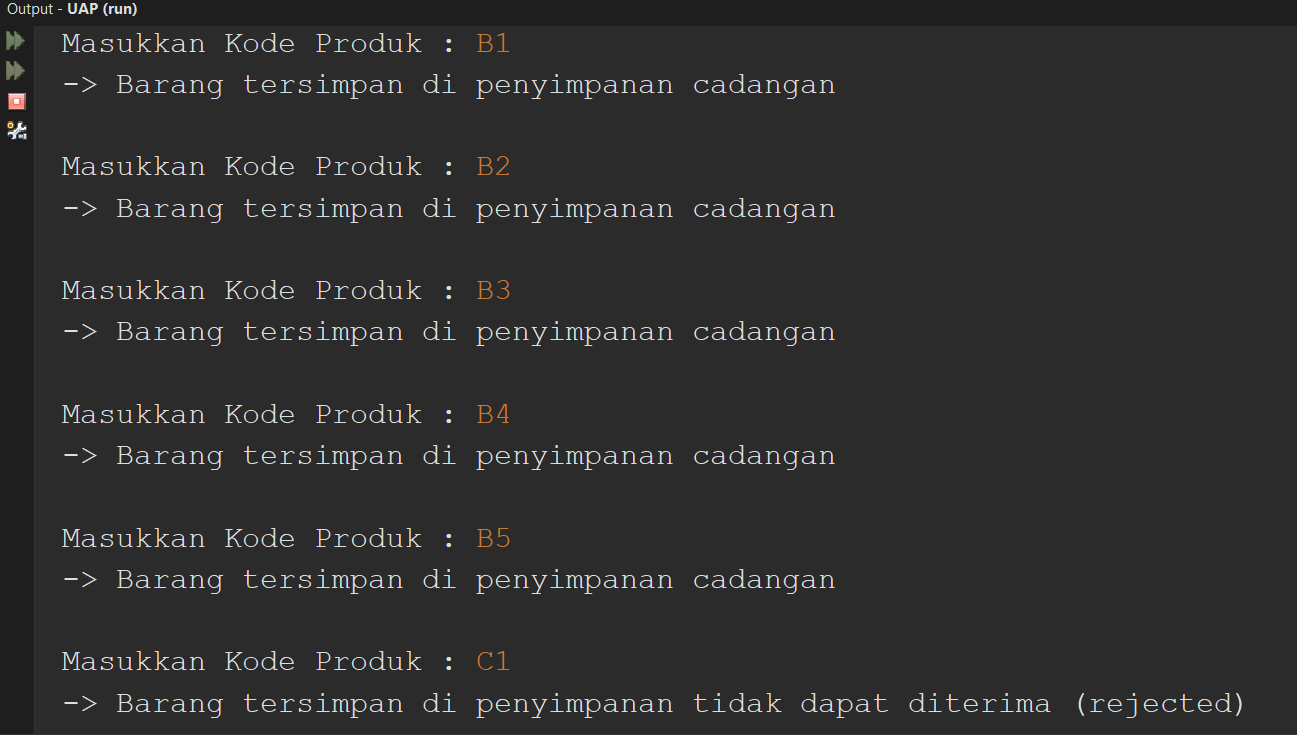
1. Main.java

|  |
| --- |
| package soal6;  import java.util.Scanner;  import java.util.Stack;  public class Main {  static Stack<String> penyimpananUtama = new Stack<>();  static Stack<String> penyimpananCadangan = new Stack<>();  static Stack<String> penyimpananRejected = new Stack<>();  static int sizePenyimpananUtama = 5, sizePenyimpananCadangan = 5, sizePenyimpananRejected = 5;  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  do {  System.out.print("Masukkan Kode Produk : ");  String kodeProduk = scan.nextLine();  simpan(kodeProduk);  } while (true);  }  static void simpan(String kode) {  if (sizePenyimpananUtama > 0) {  penyimpananUtama.add(kode);  sizePenyimpananUtama--;  System.out.println("-> Barang tersimpan di penyimpanan utama");  System.out.println("");  return;  }  if (sizePenyimpananCadangan > 0) {  penyimpananCadangan.add(kode);  sizePenyimpananCadangan--;  System.out.println("-> Barang tersimpan di penyimpanan cadangan");  System.out.println("");  return;  }  if (sizePenyimpananRejected > 0) {  penyimpananRejected.add(kode);  sizePenyimpananRejected--;  System.out.println("-> Barang tersimpan di penyimpanan tidak dapat diterima (rejected)");  System.out.println("");  }  }  } |

Output:



Gambar 6.1 hasil output



Gambar 6.2 hasil output

1. **Soal 7**

Seorang asisten dosen sedang membuat sistem penilaian berbasis HashMap yang memiliki key Nama Mahasiswa dan value Nilai Mahasiswa. Buatlah sistem penilaian tersebut dengan:

a. Jumlah nama dan skor mahasiswa minimal 10 buah (menggunakan input scanner)

b. Menampilkan nama mahasiswa yang memiliki skor terendah dan tertinggi

c. Menampilkan rata-rata skor mahasiswa

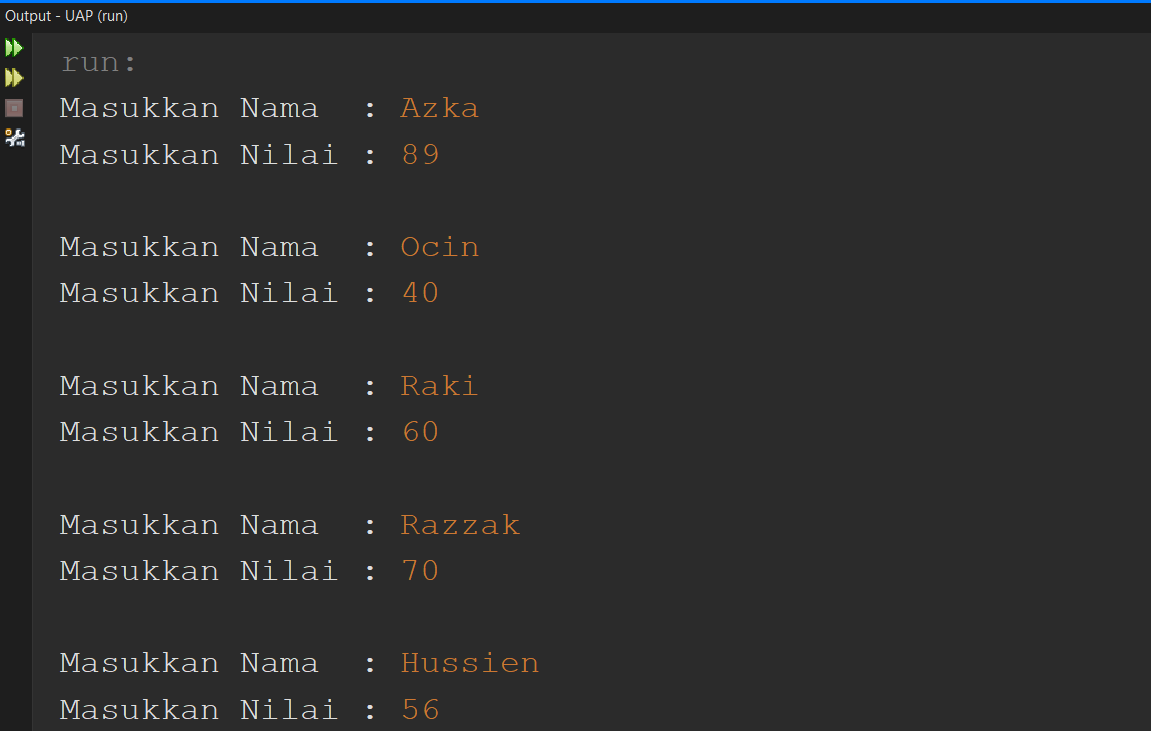
d. Menampilkan urutan mahasiswa secara ascending dan descending menggunakan salah satu algoritma sorting yang pernah dipelajari sebelumnya.

Source Code:

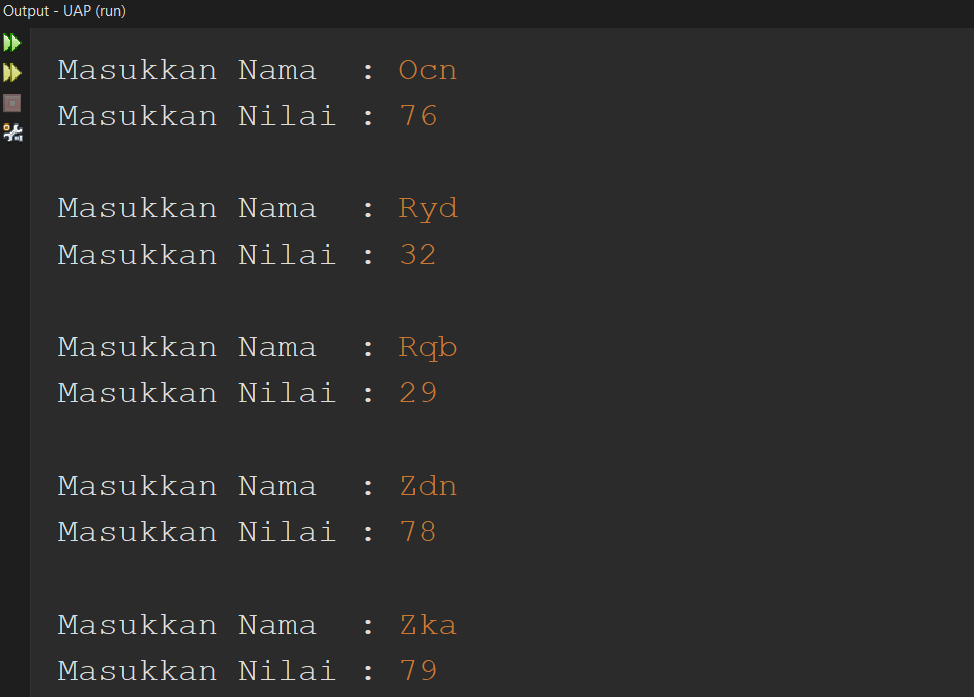
1. Main.java

|  |
| --- |
| package soal7;  import java.util.Scanner;  import java.util.HashMap;  import java.util.Map;  public class Main {  static HashMap<String, Double> penilaianMahasiswa = new HashMap<>();  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  for (int i = 0; i < 10; i++) {  System.out.print("Masukkan Nama : ");  String nama = scan.nextLine();  System.out.print("Masukkan Nilai : ");  double nilai = scan.nextDouble();  penilaianMahasiswa.put(nama, nilai);  System.out.println("");  scan = new Scanner(System.in);  }  do {  menu();  System.out.print("Masukkan Pilihan : ");  int pilihan = scan.nextInt();  switch (pilihan) {  case 1:  tampikanSkorTerendahDanTertinggi();  break;  case 2:  tampilkanRataRataSkor();  break;  case 3:  urutkanDanTampilkan();  break;  case 4:  return;  default:  System.out.println("");  }  } while (true);  }  static void menu() {  System.out.println("1. Tampilan Nama Mahasiswa yang Memiliki Skor Terendah dan Tertinggi");  System.out.println("2. Tampilkan Rata-rata Skor Mahasiswa");  System.out.println("3. Tampilkan Urutan Mahasiswa Secara Ascending dan Descending");  System.out.println("4. Keluar");  }  static void tampikanSkorTerendahDanTertinggi() {  double nilaiMin = Double.MAX\_VALUE, nilaiMax = Double.MIN\_VALUE;  String namaMin = "", namaMax = "";  for (Map.Entry<String, Double> entry : penilaianMahasiswa.entrySet()) {  if (entry.getValue() < nilaiMin) {  nilaiMin = entry.getValue();  namaMin = entry.getKey();  }  if (entry.getValue() > nilaiMax) {  nilaiMax = entry.getValue();  namaMax = entry.getKey();  }  }  System.out.println("Mahasiswa dengan skor terendah:");  System.out.println("Nama : " + namaMin);  System.out.println("Nilai: " + nilaiMin);  System.out.println("");  System.out.println("Mahasiswa dengan skor tertinggi:");  System.out.println("Nama : " + namaMax);  System.out.println("Nilai: " + nilaiMax);  System.out.println("");  }  static void tampilkanRataRataSkor() {  double rataRata = 0;  for (double nilai : penilaianMahasiswa.values()) {  rataRata += nilai;  }  rataRata /= penilaianMahasiswa.size();  System.out.println("Rata-rata skor mahasiswa: " + rataRata);  System.out.println("");  }  static void urutkanDanTampilkan() {  int sizeHashMap = penilaianMahasiswa.size();  String[] namaArray = new String[sizeHashMap];  double[] nilaiArray = new double[sizeHashMap];  int index = 0;  for (Map.Entry<String, Double> entry : penilaianMahasiswa.entrySet()) {  namaArray[index] = entry.getKey();  nilaiArray[index] = entry.getValue();  index++;  }  for (int i = 0; i < sizeHashMap - 1; i++) {  for (int j = 0; j < sizeHashMap - 1 - i; j++) {  if (nilaiArray[j] > nilaiArray[j + 1]) {  double tempNilai = nilaiArray[j];  nilaiArray[j] = nilaiArray[j + 1];  nilaiArray[j + 1] = tempNilai;  String tempNama = namaArray[j];  namaArray[j] = namaArray[j + 1];  namaArray[j + 1] = tempNama;  }  }  }  System.out.println("Urutan mahasiswa berdasarkan nilai (Ascending) :");  for (int i = 0; i < sizeHashMap; i++) {  System.out.println((i + 1) + ". Nama: " + namaArray[i] + ", Nilai: " + nilaiArray[i]);  }  System.out.println("");  for (int i = 0; i < sizeHashMap - 1; i++) {  for (int j = 0; j < sizeHashMap - 1 - i; j++) {  if (nilaiArray[j] < nilaiArray[j + 1]) {  double tempNilai = nilaiArray[j];  nilaiArray[j] = nilaiArray[j + 1];  nilaiArray[j + 1] = tempNilai;  String tempNama = namaArray[j];  namaArray[j] = namaArray[j + 1];  namaArray[j + 1] = tempNama;  }  }  }  System.out.println("Urutan mahasiswa berdasarkan nilai (Descending) :");  for (int i = 0; i < sizeHashMap; i++) {  System.out.println((i + 1) + ". Nama: " + namaArray[i] + ", Nilai: " + nilaiArray[i]);  }  System.out.println("");  }  } |

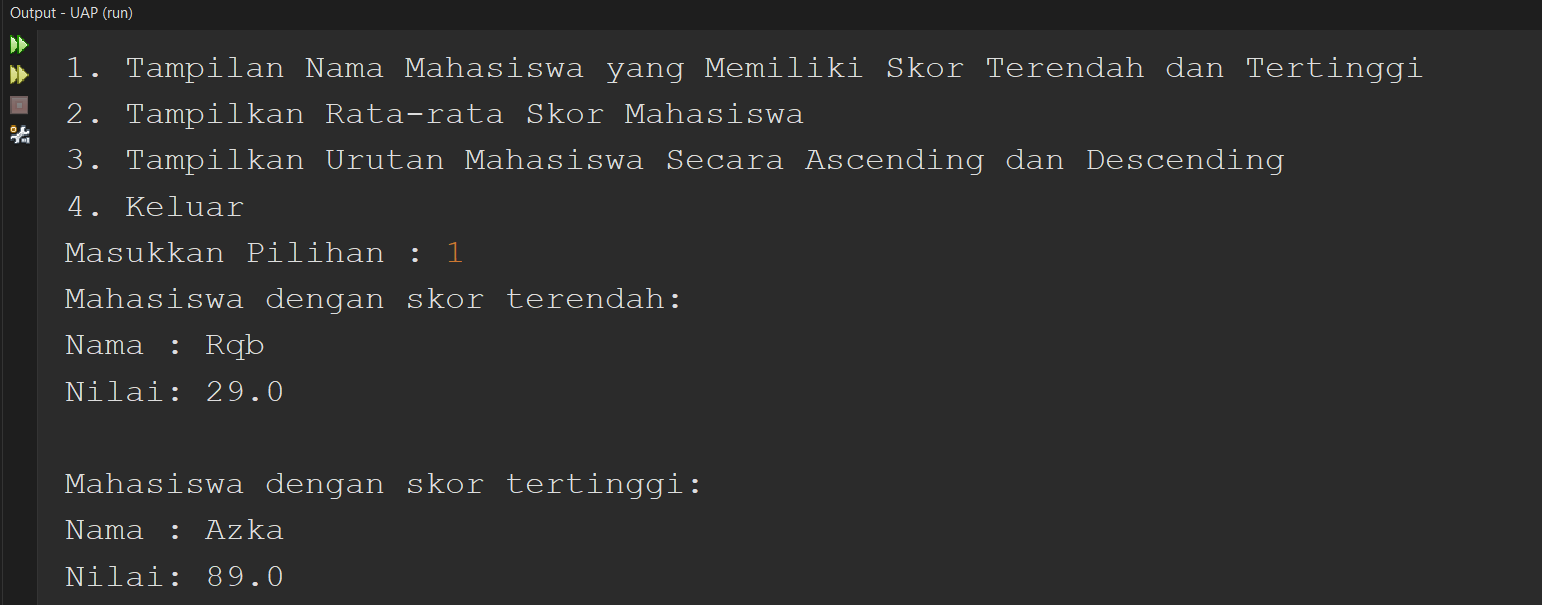
Output:



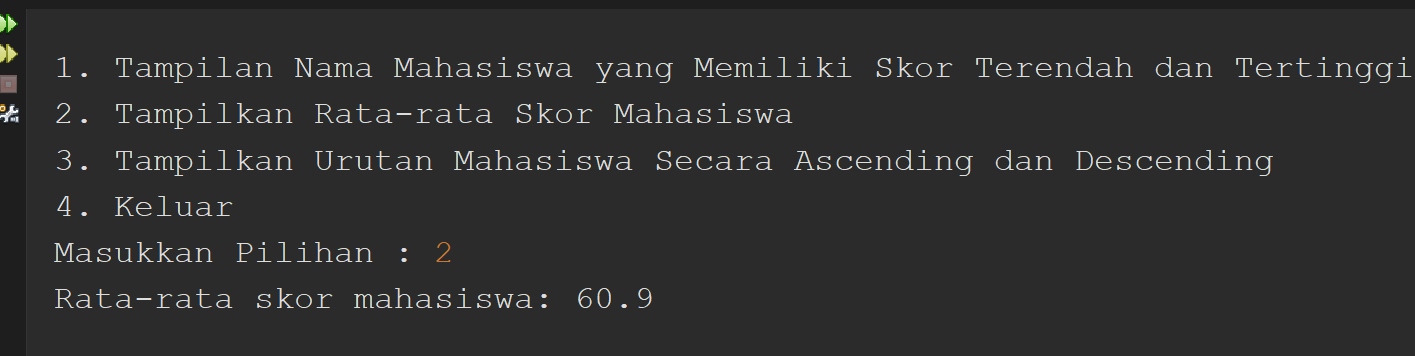
Gambar 7.1 hasil output



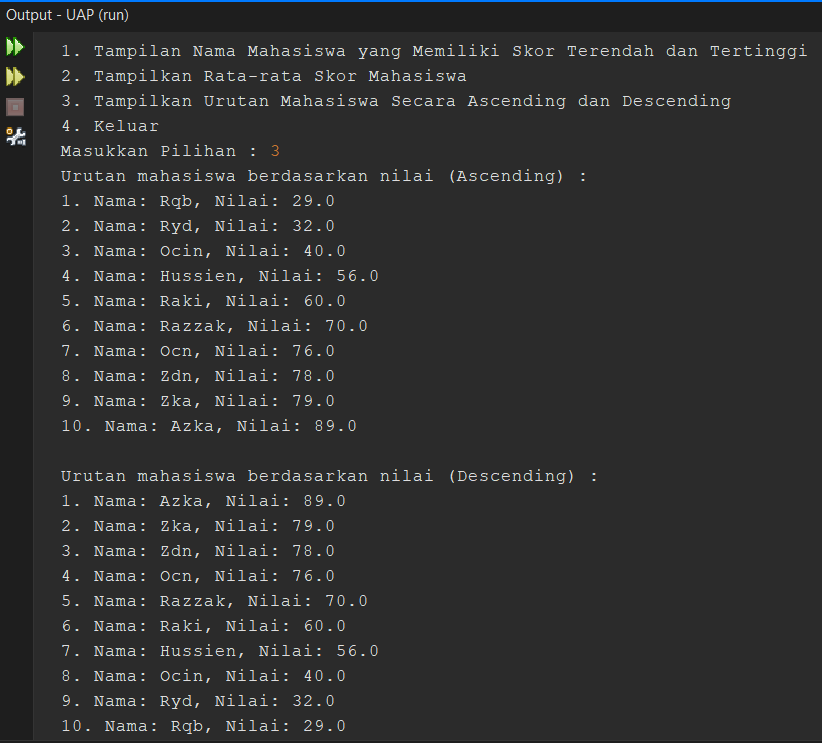
Gambar 7.2 hasil output



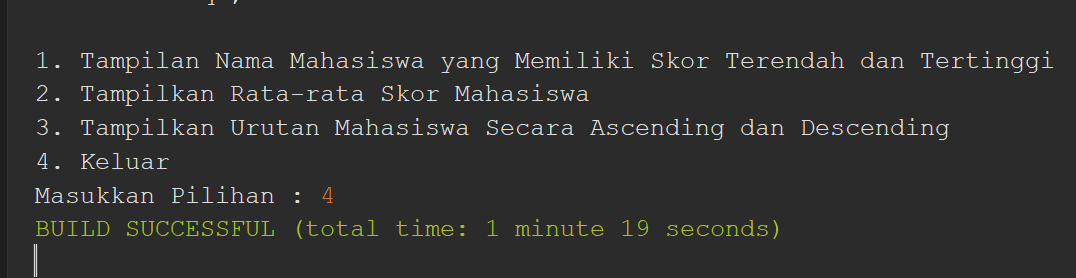
Gambar 7.3 hasil output



Gambar 7.4 hasil output



Gambar 7.5 hasil output



Gambar 7.6 hasil output

1. **Soal 8**

Buatlah fungsi untuk melakukan validasi anagram (kedua string harus memiliki huruf yang sama dalam jumlah yang sama; lebih lengkapnya silahkan cek Google) dan fungsi untuk menghitung jumlah kemunculan karakter pada dua buah string) pada nama Anda, tanpa spasi menggunakan struktur data HashMap, misalkan: “muhammadrizan”. Selanjutnya, lakukan langkah berikut:

a. Pertama, anda memiliki string dari nama anda yang didefinisikan oleh variabel nama. Selanjutnya, buatlah variabel reversed\_nama dan lakukan reverse pada variabel nama

b. Lakukan validasi anagram pada variabel nama dan reversed\_nama dan output yang diharapkan antara true atau false

c. Selanjutnya, hitung frekuensi karakter dari kedua variabel tersebut

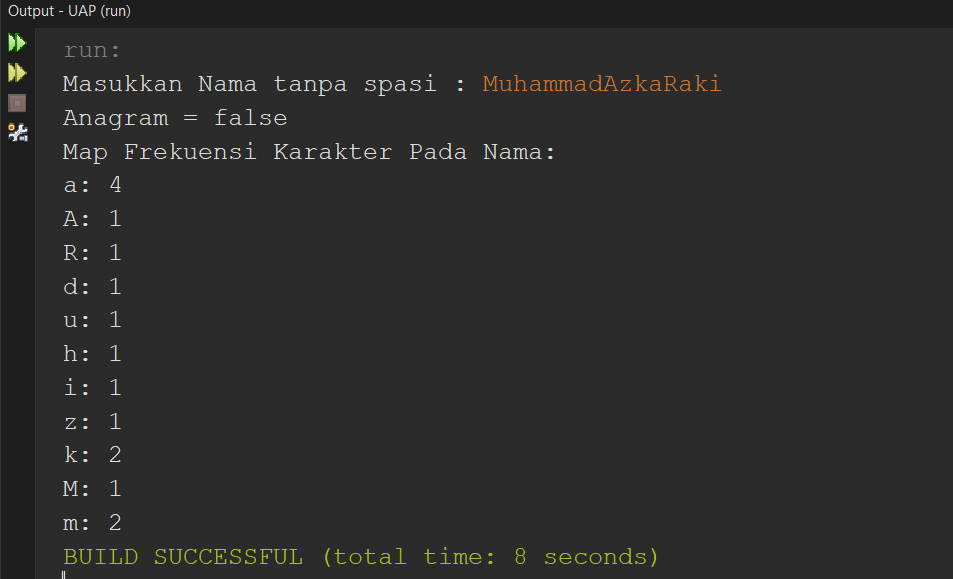
Misalnya: m:3, u:1, h:1, a:3, d:1, r:1, i:1, z:1, n:1

Source Code:

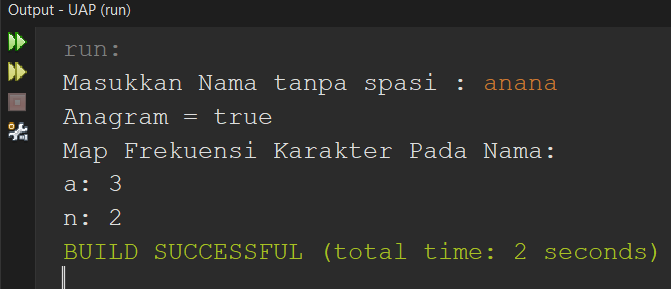
1. Main.java

|  |
| --- |
| package soal8;  import java.util.Scanner;  import java.util.HashMap;  import java.util.Map;  public class Main {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  System.out.print("Masukkan Nama tanpa spasi : ");  String nama = scan.nextLine();  String reversed\_nama = reverseNama(nama);  System.out.println("Anagram = " + nama.equals(reversed\_nama));  HashMap<Character, Integer> frekuensiKarakter = new HashMap<>();  for (int i = 0; i < nama.length(); i++) {  if (frekuensiKarakter.containsKey(nama.charAt(i))) {  frekuensiKarakter.put(nama.charAt(i), frekuensiKarakter.get(nama.charAt(i)) + 1);  } else {  frekuensiKarakter.put(nama.charAt(i), 1);  }  }  System.out.println("Map Frekuensi Karakter Pada Nama: ");  for (Map.Entry<Character, Integer> entry : frekuensiKarakter.entrySet()) {  System.out.println(entry.getKey() + ": " + entry.getValue());  }  }  static String reverseNama(String nama) {  String str = "";  for (int i = nama.length() - 1; i >= 0; i--) {  str += nama.charAt(i);  }  return str;  }  } |

Output:



Gambar 8.1 hasil output



Gambar 8.2 hasil output