

## MODUL 3

No.	Soal
1	<p>Apakah perbedaan antara struktur kontrol percabangan if-else dan switch-case?</p> <p><b>Jawaban</b></p> <ol style="list-style-type: none"> <li> <p><b>Penggunaan kondisi</b></p> <p><b>if-else:</b> Dalam struktur if-else, kondisi dievaluasi secara berurutan. Jika kondisi pertama benar, blok kode yang terkait dengan itu dieksekusi, jika tidak, maka kondisi berikutnya akan dievaluasi.</p> <p><b>switch-case:</b> Dalam switch-case, ekspresi yang dievaluasi hanya sekali, dan hasilnya digunakan untuk memilih salah satu dari beberapa opsi. Setelah opsi yang sesuai ditemukan, blok kode yang terkait dengan itu dieksekusi.</p> </li> <li> <p><b>Kondisi yang dievaluasi</b></p> <p><b>if-else:</b> Kondisi dalam if-else bisa sangat fleksibel dan kompleks, dapat berupa ekspresi boolean apa pun.</p> <p><b>switch-case:</b> Ekspresi yang dievaluasi dalam switch-case biasanya harus berupa nilai tunggal (biasanya integer, karakter, atau enumerasi). Dalam beberapa bahasa pemrograman, switch-case mungkin juga memungkinkan penggunaan tipe data string.</p> </li> <li> <p><b>Jumlah cabang</b></p> <p><b>if-else:</b> Tidak ada batasan pada jumlah cabang if-else yang dapat digunakan dalam satu blok if-else. Anda dapat menambahkan sebanyak mungkin kondisi yang diperlukan.</p> <p><b>switch-case:</b> Jumlah cabang dalam switch-case biasanya terbatas dan tetap statis selama kompilasi. Biasanya, hanya satu ekspresi yang dapat dievaluasi untuk memilih antara beberapa kasus.</p> </li> <li> <p><b>Kejelasan kode</b></p> <p><b>if-else:</b> Cocok digunakan ketika ada banyak kondisi yang saling berkaitan dan tidak dapat dikelompokkan dengan jelas.</p> <p><b>switch-case:</b> Lebih cocok digunakan ketika ekspresi yang dievaluasi memiliki banyak pilihan tetap yang harus dipertimbangkan.</p> </li> <li> <p><b>Kemungkinan kesalahan</b></p> <p><b>if-else :</b> Karena setiap kondisi dievaluasi secara berurutan, kesalahan mungkin terjadi jika kondisi yang benar terlewatkan atau ada kondisi yang saling bertentangan.</p> <p><b>switch-case:</b> Lebih tidak mungkin terjadi kesalahan dalam switch-case karena ekspresi hanya dievaluasi sekali, dan setiap kasus ditangani secara eksplisit.</p> <p>Pemilihan antara if-else dan switch-case tergantung pada kompleksitas kondisi, kejelasan kode, dan jenis data yang dievaluasi. Meskipun keduanya memiliki peran yang serupa dalam mengarahkan aliran program, pilihan yang tepat tergantung pada konteks dan kebutuhan spesifik dari tugas pemrograman yang sedang dihadapi.</p> </li> </ol>
2.	<p><b>Soal</b></p> <p>Kapan digunakan struktur kontrol if-else dan switch-case</p> <p><b>Jawaban</b></p> <p><b>Penggunaan If-Else:</b></p>

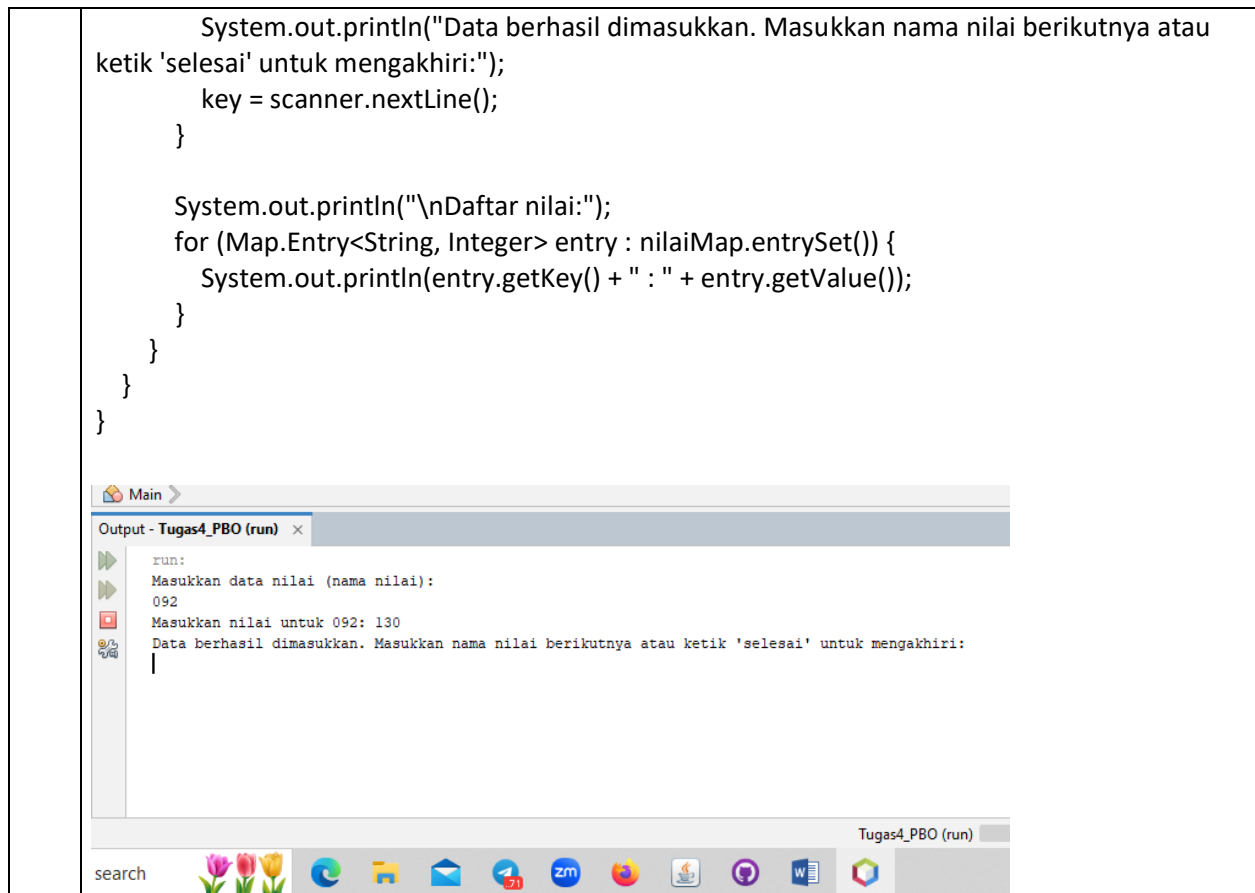
	<p>1. <b>Kondisi yang kompleks:</b> Jika kondisi yang dievaluasi sangat kompleks dan tidak dapat direpresentasikan dengan baik oleh ekspresi tunggal atau kasus-kasus yang terbatas.</p> <p>2. <b>Kondisi bersarang:</b> Ketika kondisi-kondisi tersebut memiliki tingkat kecermatan yang berbeda dan bisa bersarang di dalam satu sama lain.</p> <p>3. <b>Kondisi saling tergantung:</b> Jika terdapat ketergantungan antara kondisi-kondisi, yang artinya satu kondisi harus dievaluasi berdasarkan hasil dari kondisi lainnya.</p> <p>4. <b>Fleksibilitas:</b> Jika Anda memerlukan fleksibilitas penuh dalam menangani berbagai macam kondisi dan kasus yang tidak dapat diprediksi sebelumnya.</p> <p><b>Penggunaan Switch-Case:</b></p> <p>1. <b>Kondisi dengan banyak kasus tetap:</b> Jika Anda memiliki sebuah ekspresi yang memiliki banyak pilihan tetap dan terbatas.</p> <p>2. <b>Ekspresi sederhana:</b> Jika ekspresi yang dievaluasi adalah jenis data sederhana seperti integer, karakter, atau enumerasi.</p> <p>3. <b>Kode yang jelas:</b> Ketika Anda ingin membuat kode yang lebih mudah dibaca dan dipahami oleh orang lain, terutama jika kasus-kasusnya cukup jelas dan terbatas.</p> <p>4. <b>Kinerja:</b> Dalam beberapa bahasa pemrograman, switch-case dapat memberikan performa yang lebih baik daripada serangkaian if-else yang panjang, terutama ketika ada banyak kondisi.</p> <p>5. <b>Pemilihan opsi:</b> Jika Anda perlu memilih salah satu dari beberapa opsi tetap berdasarkan hasil ekspresi, seperti dalam penggunaan menu atau pemilihan opsi permainan.</p> <p>Dalam banyak kasus, keputusan untuk menggunakan if-else atau switch-case juga tergantung pada preferensi pribadi dan gaya penulisan kode dalam tim pengembangan. Namun, memahami karakteristik dan kegunaan masing-masing dapat membantu membuat keputusan yang lebih tepat.</p>
3.	<p><b>Soal</b></p> <p>Pada program 2, tambahkan perintah untuk memilih 2 opsi menggunakan kontrol switch..case. opsi pilihah 1=inputNilai() Pilihan 2=inputNilaiBaru()</p> <p><b>Jawaban</b></p> <pre>package pertemuan2.modul3.nilai;  import java.util.Scanner;  public class TestNilai {     public static void main(String[] args) {         HitungRata hitung = new HitungRata();         Scanner input = new Scanner(System.in);         int pilihan;          do {</pre>

	<pre> System.out.println("Menu:"); System.out.println("1. Input Nilai"); System.out.println("2. Input Nilai Baru"); System.out.println("0. Keluar"); System.out.print("Pilih opsi: "); pilihan = input.nextInt();  switch (pilihan) {     case 1:         System.out.print("Masukkan Jumlah Data : ");         int banyakData = input.nextInt();         int nilai[] = new int[banyakData];         System.out.print("Masukkan Nilai : ");         hitung.inputNilai(nilai);         System.out.print("Daftar Nilai : ");         hitung.cetakNilai(nilai);         System.out.println("Rata Nilai : " + hitung.rataNilai(banyakData));         break;     case 2:         System.out.print("Masukkan Nilai Baru: ");         hitung.inputNilaiBaru();         System.out.print("Daftar Nilai Baru : ");         hitung.cetakNilaiBaru();         break;     case 0:         System.out.println("Keluar dari program.");         break;     default:         System.out.println("Opsi tidak valid. Silakan pilih opsi yang benar.");         break; } } while (pilihan != 0); input.close(); } } </pre>
4.	Soal
	Apakah perbedaan antara struktur kontrol perulangan while dan do-while?
	Jawaban
	<p>Struktur kontrol perulangan while dan do-while adalah dua cara yang berbeda untuk mengulang eksekusi blok kode berdasarkan kondisi tertentu. Meskipun keduanya memiliki fungsi yang mirip, ada perbedaan utama antara keduanya:</p> <p><b>Evaluasi kondisi:</b></p> <p><b>while:</b> Kondisi dievaluasi sebelum blok kode dieksekusi. Jika kondisi awalnya salah, maka blok kode tidak akan pernah dieksekusi.</p> <p><b>do-while:</b> Blok kode dieksekusi sekali sebelum kondisi dievaluasi. Artinya, setidaknya satu iterasi akan selalu dilakukan sebelum pengecekan kondisi.</p>

	<p><b>Urutan eksekusi:</b></p> <p><b>while:</b> Kondisi dievaluasi sebelum eksekusi blok kode, sehingga ada kemungkinan bahwa blok kode tidak akan pernah dieksekusi sama sekali jika kondisi awalnya salah.</p> <p><b>do-while:</b> Blok kode dieksekusi sekali sebelum pengecekan kondisi, sehingga setidaknya satu iterasi akan selalu dilakukan bahkan jika kondisi awalnya salah.</p> <p><b>Kapan digunakan:</b></p> <p><b>while:</b> Digunakan ketika tidak jelas apakah blok kode harus dieksekusi minimal satu kali, atau mungkin tidak perlu dieksekusi sama sekali tergantung pada kondisi.</p> <p><b>do-while:</b> Digunakan ketika Anda ingin memastikan bahwa blok kode dieksekusi setidaknya satu kali, bahkan jika kondisi awalnya salah.</p> <p><b>Kejelasan kode:</b></p> <p><b>do-while:</b> Kadang-kadang dapat meningkatkan kejelasan kode karena memastikan bahwa setidaknya satu iterasi dilakukan, yang dapat berguna dalam situasi-situasi di mana setiap iterasi penting.</p> <p><b>while:</b> Lebih jelas dalam kasus di mana tidak ada kebutuhan untuk menjamin setidaknya satu iterasi.</p> <p><b>Kasus penggunaan:</b></p> <p><b>do-while:</b> Berguna dalam skenario di mana Anda perlu membaca masukan dari pengguna setidaknya sekali sebelum melakukan pengecekan kondisi.</p> <p><b>while:</b> Cocok untuk banyak kasus pengulangan yang umum di mana kondisi awal yang salah mungkin terjadi, dan tidak ada kebutuhan untuk menjamin setidaknya satu iterasi.</p> <p>Dalam pemilihan antara while dan do-while, penting untuk mempertimbangkan apakah Anda ingin memastikan setidaknya satu iterasi dilakukan atau tidak, serta jika blok kode harus dieksekusi sebelum atau setelah pengecekan kondisi.</p>
5.	Soal
	Kapan digunakan struktur kontrol for?
	Jawaban
	<p>Struktur kontrol <b>for</b> digunakan ketika Anda memiliki kebutuhan untuk melakukan iterasi (pengulangan) sejumlah tertentu kali, dengan batasan dan langkah yang sudah ditentukan sebelumnya. Ini sangat berguna ketika Anda ingin menjalankan suatu blok kode secara berulang dengan jumlah iterasi yang jelas. Berikut adalah beberapa kasus umum di mana struktur kontrol <b>for</b> sangat berguna:</p> <p><b>Iterasi pada rentang nilai tertentu:</b> Ketika Anda perlu melakukan iterasi pada rentang nilai tertentu, misalnya dari 1 hingga 10, atau dari 0 hingga 100.</p> <p><b>Akses ke elemen array atau koleksi:</b> <b>for</b> loop sering digunakan untuk mengakses elemen-elemen dalam array atau koleksi dengan cara yang terstruktur dan efisien.</p>

	<p><b>Perulangan berbasis hitungan:</b> Misalnya, ketika Anda ingin melakukan iterasi sejumlah tertentu kali untuk melakukan operasi tertentu seperti penghitungan atau pengolahan data.</p> <p><b>Iterasi menggunakan langkah tertentu:</b> Anda dapat menentukan langkah iterasi dalam pernyataan for, sehingga Anda dapat melakukan iterasi dengan penambahan atau pengurangan tertentu pada setiap langkah iterasi.</p> <p><b>Perulangan bersarang:</b> for loop juga berguna dalam kasus perulangan bersarang, di mana Anda memiliki lebih dari satu loop yang harus dieksekusi bersamaan.</p> <p><b>Iterasi dalam algoritma pengulangan:</b> Dalam beberapa algoritma seperti algoritma pencarian, pengurutan, atau transformasi data, struktur kontrol for sering digunakan untuk mengatur proses iterasi.</p>
6.	<p><b>Soal</b></p> <p>Apakah perbedaan antara Array dan ArrayList?berilah contoh masing-masing!</p> <p><b>Jawaban</b></p> <p>Perbedaan utama antara Array dan ArrayList adalah dalam fleksibilitas dan sifat dinamisnya:</p> <p><b>Array:</b></p> <ul style="list-style-type: none"> <li>- Array adalah struktur data statis yang memiliki ukuran tetap yang ditentukan saat pembuatan array.</li> <li>- Setelah array dibuat, ukurannya tidak dapat diubah.</li> <li>- Array hanya dapat menyimpan elemen dengan tipe data yang sama.</li> <li>- Akses elemen dalam array dilakukan menggunakan indeks.</li> <li>- Dalam Java, array dapat dideklarasikan menggunakan sintaks seperti berikut:</li> </ul> <pre>java int[] numbers = new int[5]; // Mendeklarasikan array integer dengan panjang 5</pre> <p><b>ArrayList:</b></p> <ul style="list-style-type: none"> <li>- ArrayList adalah bagian dari framework Collections dalam Java, yang menyediakan struktur data dinamis.</li> <li>- Ukuran ArrayList dapat berubah secara dinamis saat program berjalan.</li> <li>- ArrayList dapat menyimpan elemen dengan tipe data yang berbeda.</li> <li>- Akses elemen dalam ArrayList juga menggunakan indeks, tetapi fungsi-fungsi tambahan seperti `add()`, `remove()`, dan `get()` digunakan untuk manipulasi data.</li> <li>- Dalam Java, ArrayList dapat dideklarasikan menggunakan sintaks seperti berikut:</li> </ul> <pre>java import java.util.ArrayList;  ... ArrayList&lt;Integer&gt; numbers = new ArrayList&lt;Integer&gt;(); // Mendeklarasikan ArrayList yang berisi bilangan bulat</pre> <p><b>Contoh penggunaan Array:</b></p> <pre>java int[] numbers = new int[5]; // Deklarasi array integer dengan panjang 5</pre>

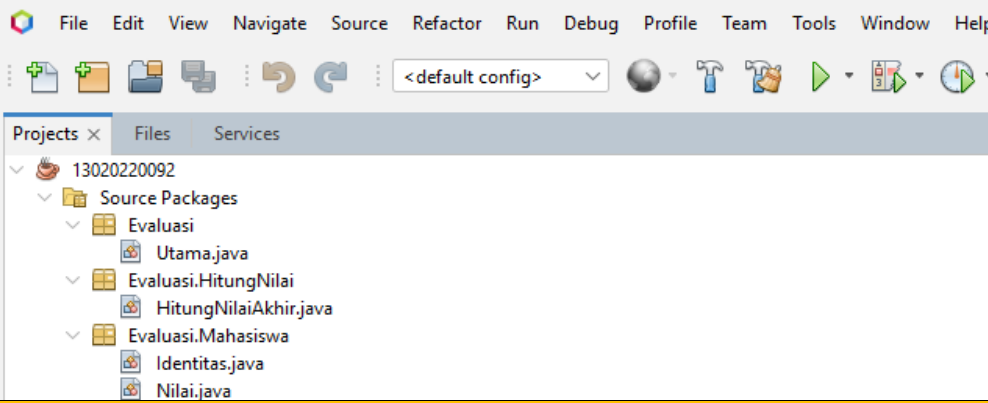
	<pre> numbers[0] = 10; numbers[1] = 20; numbers[2] = 30; numbers[3] = 40; numbers[4] = 50;  for (int i = 0; i &lt; numbers.length; i++) {     System.out.println(numbers[i]); } </pre> <p><b>Contoh penggunaan ArrayList:</b></p> <pre> java import java.util.ArrayList; ArrayList&lt;Integer&gt; numbers = new ArrayList&lt;Integer&gt;(); // Deklarasi ArrayList yang berisi bilangan bulat numbers.add(10); numbers.add(20); numbers.add(30); numbers.add(40); numbers.add(50);  for (int i = 0; i &lt; numbers.size(); i++) {     System.out.println(numbers.get(i)); } </pre>
7.	<p><b>Soal</b></p> <p>Buatlah contoh program yang mengimplementasikan HashMap dengan memasukkan nilai dan key melalui keyboard!</p> <p><b>Jawaban</b></p> <pre> import java.util.HashMap; import java.util.Map; import java.util.Scanner;  public class Main {     public static void main(String[] args) {         try (Scanner scanner = new Scanner(System.in)) {             Map&lt;String, Integer&gt; nilaiMap = new HashMap&lt;&gt;();              System.out.println("Masukkan data nilai (nama nilai:");             String key = scanner.nextLine();             while (!key.equals("selesai")) {                 System.out.print("Masukkan nilai untuk " + key + ": ");                 int nilai = scanner.nextInt();                 scanner.nextLine(); // Membuang karakter newline (\n) dari input sebelumnya                  nilaiMap.put(key, nilai);             }         }     } } </pre>



MODUL 4	
No.	Soal
1.	Berdasarkan ke tiga program di atas Class utama, Class Orang dan Class Mahasiswa, manakah yang menunjukkan konsep pewarisan dan polimorfisme! Jelaskan konsep tersebut sesuai program tersebut!
	Jawaban
	<b>Pewarisan (Inheritance):</b> <pre> public class Mahasiswa extends Orang {     private String stb;     public Mahasiswa() {         super(); // Memanggil konstruktor kelas Orang         this.stb = "1302002134"; // stambuk anda     } } </pre>



	<p>Dalam Class Mahasiswa, kita melihat bahwa konstruktor tanpa argumen Mahasiswa() menggunakan kata kunci super() untuk memanggil konstruktor kelas induknya, yaitu Orang(). Ini menunjukkan bahwa Mahasiswa mewarisi konstruktor dari kelas Orang.</p> <p>Dalam metode main di kelas Utama, kita membuat objek dari kelas Orang dan mengakses atributnya. Meskipun objek yang dibuat adalah objek dari kelas Orang, konstruktor dari kelas Mahasiswa tidak dipanggil.</p> <p><b>Polimorfisme:</b></p> <p><b>Polimorfisme tidak terjadi dalam program ini karena tidak ada metode yang di-override atau diimplementasikan ulang dalam kelas turunan yang memiliki nama yang sama dengan metode di kelas induk. Polimorfisme umumnya terjadi ketika metode yang sama diimplementasikan ulang dengan perilaku yang berbeda dalam kelas turunan.</b></p>
2.	Soal
	<p>Tambahkan static pada method info() Class Orang dan Class Mahasiswa kemudian lakukan pemanggilan method info() pada program utama (Class utama)!</p>
	Jawaban
	<pre> public class Orang {     public String nama;     public Orang() {         this.nama= "Dzaky";     }     public Orang(String nama) {         this.nama = nama;     }     public static void info() {         System.out.println("Ini adalah info dari kelas Orang.");     } } public class Mahasiswa extends Orang {     private String stb;     public Mahasiswa() {         super();         this.stb="1302002134";     }     public Mahasiswa(String stb, String nama) {         this.nama = nama;         this.stb = stb;     }     public static void info() {         System.out.println("Ini adalah info dari kelas Mahasiswa.");     } } public class Utama {     public static void main(String[] args) { </pre>



	<pre> Orang.info(); // Memanggil static method info() dari kelas Orang Mahasiswa.info(); // Memanggil static method info() dari kelas Mahasiswa } } </pre>
3.	Soal
	Buatlah sebuah project dengan nama project stambuk anda dan buatlah pengorganisasian package dan class seperti berikut
	Jawaban
	
	Setelah mengerjakan soal nomor 3, Lengkapi Program berikut
	<h3>Package Evaluasi.Mahasiswa</h3> <pre> package EvaluasiMahasiswa; public class Identitas {     private String nama;     private String stambuk;      public void setName(String nama) {         this.nama = nama;     }      public String getName() {         return this.nama;     }      public void setStambuk(String stambuk) {         this.stambuk = stambuk;     }      public String getStambuk() {         return this.stambuk;     } } </pre>

	<pre> package EvaluasiMahasiswa;  public class Nilai {     private int tugas1;     private int tugas2;     private int mid;     private int uas;      public void setTugas1(int tugas1) {         this.tugas1 = tugas1;     }     public int getTugas1() {         return this.tugas1;     }     public void setTugas2(int tugas2) {         this.tugas2 = tugas2;     }     public int getTugas2() {         return this.tugas2;     }     public void setMid(int mid) {         this.mid = mid;     }     public int getMid() {         return this.mid;     }     public void setUas(int uas) {         this.uas = uas;     }     public int getUas() {         return this.uas;     } } </pre>
Package Evaluasi.HitungNilai	
	<pre> package EvaluasiHitungNilai;  public class HitungNilaiAkhir {     public double nilaiTugas(int tugas1, int tugas2) {         return (tugas1 + tugas2) / 2.0;     }     public double nilaiAkhir(double tugas, int mid, int uas) {         return (tugas * 0.4) + (mid * 0.3) + (uas * 0.3);     } } </pre>

	<pre>     } } </pre>
Package Evaluasi	
	<pre> package Evaluasi;  import EvaluasiHitungNilai.HitungNilaiAkhir; import EvaluasiMahasiswa.Identitas; import EvaluasiMahasiswa.Nilai;  import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; import java.util.Scanner; import javax.swing.JOptionPane;  public class Utama {     public static void main(String[] args) throws IOException {         Scanner scanner = new Scanner(System.in);         BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));          System.out.print("Masukkan Nama: ");         String nama = scanner.nextLine();         System.out.print("Masukkan Stambuk: ");         String stambuk = scanner.nextLine();          System.out.print("Masukkan Nilai Tugas 1: ");         int tugas1 = Integer.parseInt(bufferedReader.readLine());         System.out.print("Masukkan Nilai Tugas 2: ");         int tugas2 = Integer.parseInt(bufferedReader.readLine());         System.out.print("Masukkan Nilai MID: ");         int mid = Integer.parseInt(bufferedReader.readLine());         System.out.print("Masukkan Nilai UAS: ");         int uas = Integer.parseInt(bufferedReader.readLine());          Identitas identitas = new Identitas();         Nilai nilai = new Nilai();          HitungNilaiAkhir hitungNilai = new HitungNilaiAkhir();         double tugas = hitungNilai.nilaiTugas(tugas1, tugas2);          double na = hitungNilai.nilaiAkhir(tugas, mid, uas);          JOptionPane.showMessageDialog(null, "Nama: " + nama + "\nStambuk: " + stambuk);         JOptionPane.showMessageDialog(null, "Nilai Tugas: " + tugas + "\nNilai MID: " + mid + "\nNilai UAS: " + uas + "\nNilai Akhir: " + na); </pre>

	<pre>} }</pre>
Output	
	<div data-bbox="292 325 1312 959"><p>Output - 13020220092 (run)</p><div> run:  Masukkan nama: Ahmad Dzaki Ubaidillah  Masukkan stambuk: 13020220992  Masukkan nilai tugas 1: 90 Masukkan nilai tugas 2: 86 Masukkan nilai mid: 93 Masukkan nilai uas: 88 Nama: Ahmad Dzaki Ubaidillah Stambuk: 13020220992 Nilai Akhir: 89.5 BUILD SUCCESSFUL (total time: 1 minute 8 seconds)</div></div>