**Nama : Ainur Rohman Hardini**

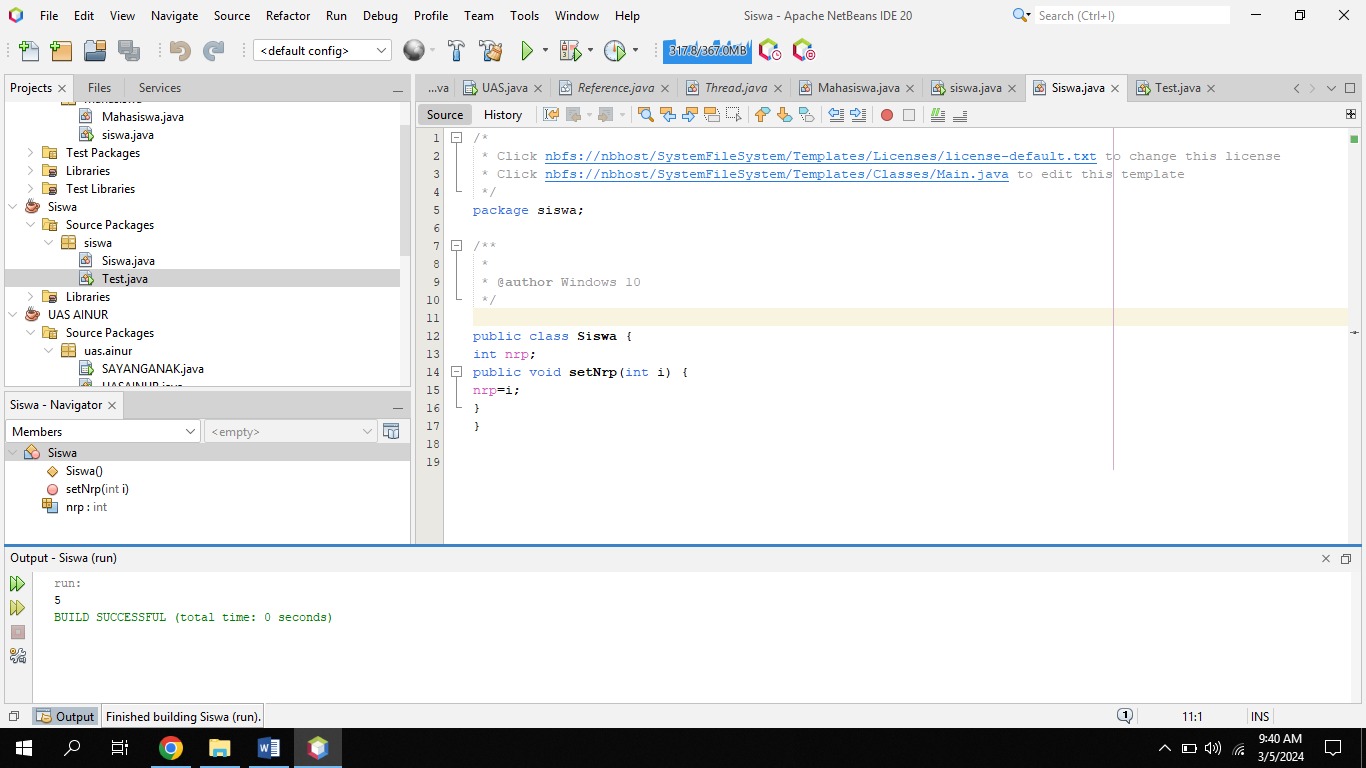
**Nim : 4123016**

**Kelas : B semester 2**

**Mata Kuliah : Alpro2**

**D. PERCOBAAN**

**1. Mengakses data member suatu class.**



**Public class namasiswa {**

**int nrp;**

**}**

**public class siswa {**

**public static void main(String args[]) {**

**namasiswa anak=new namasiswa();**

**anak.nrp = 5 ;**

**System.out.println(anak.nrp);**

**}**

**}**

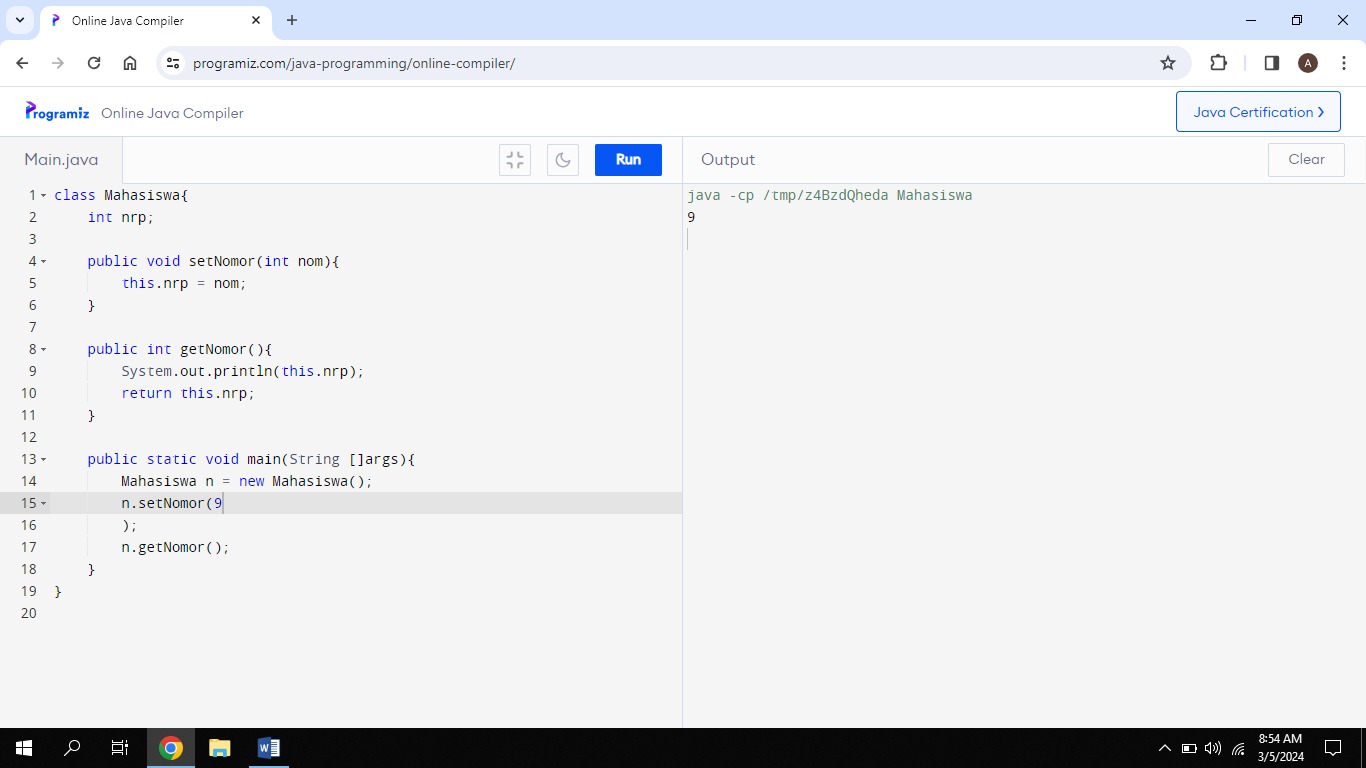
**Analisis:**

**Program di atas mencoba membuat objek dari kelas namasiswa dan kemudian mencetak nilai variabel nrp.**

**Kesimpulan:**

**Program di atas mendemonstrasikan cara mendefinisikan kelas, variabel, dan konstruktor.**

**2. Mengakses method suatu class.**



**class Mahasiswa{**

**int nrp;**

**public void setNomor(int nom){**

**this.nrp = nom;**

**}**

**public int getNomor(){**

**System.out.println(this.nrp);**

**return this.nrp;**

**}**

**public static void main(String []args){**

**Mahasiswa n = new Mahasiswa();**

**n.setNomor(5**

**);**

**n.getNomor();**

**}**

**}**

**Analisis:**

**Definisi Kelas:**

* **Kode mendefinisikan sebuah kelas bernama Mahasiswa.**
* **Kelas ini memiliki atribut bernama nrp bertipe int, yang kemungkinan mewakili nomor registrasi pokok mahasiswa.**

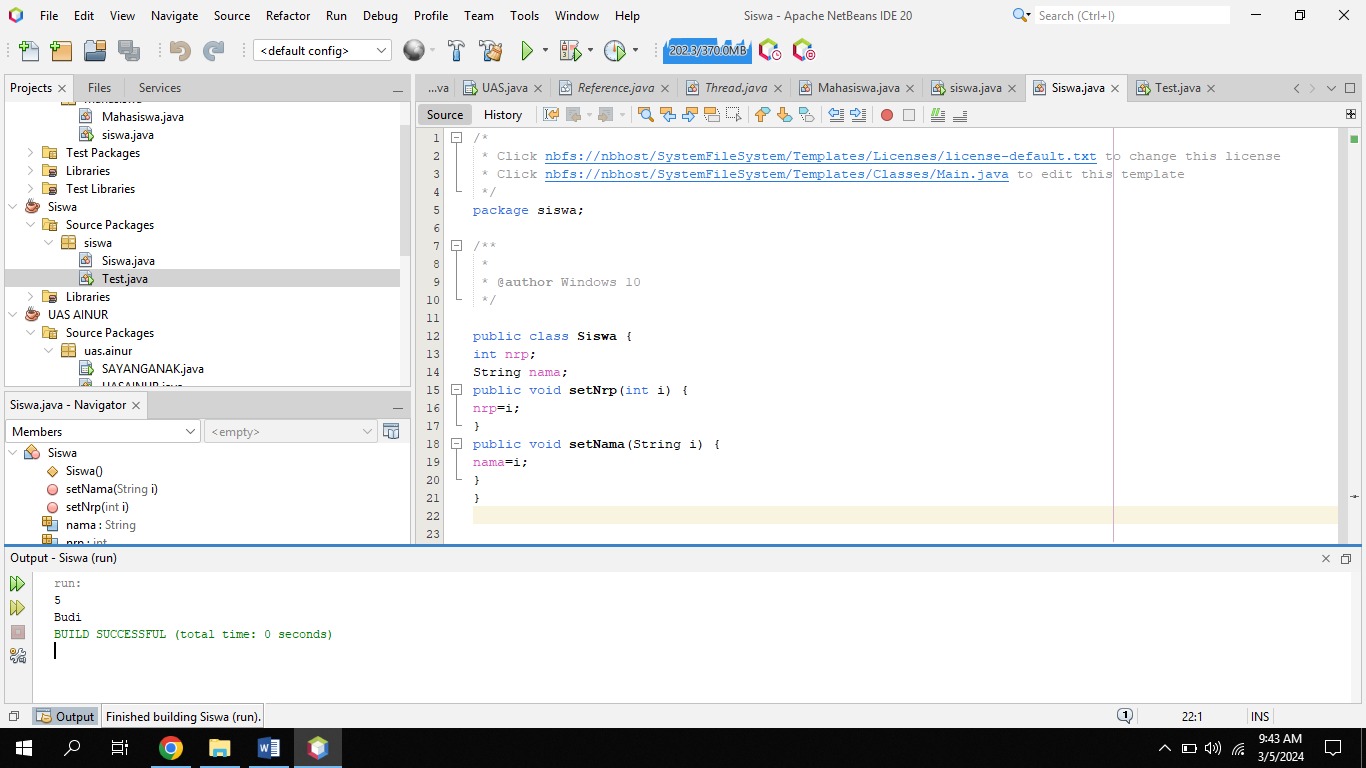
**Metode:**

* **setNomor(int nom):**
  + **Metode publik ini menerima parameter integer nom.**
  + **Metode ini menetapkan nilai nom ke atribut nrp menggunakan kata kunci this untuk merujuk ke variabel instance objek saat ini.**
* **getNomor():**
  + **Metode publik ini tidak menerima argumen apa pun.**
  + **Metode ini mencetak nilai atribut nrp ke konsol menggunakan System.out.println.**
  + **Kemudian mengembalikan nilai atribut nrp, yang mungkin berlebihan karena sudah dicetak.**

**Metode Main:**

* **Metode main adalah titik awal program.**
* **Metode ini membuat instance baru dari kelas Mahasiswa bernama n.**
* **Memanggil metode setNomor pada objek n, yang menetapkan nilai nrp menjadi 5.**
* **Kemudian memanggil metode getNomor pada objek n, yang mencetak "5" ke konsol dan mengembalikan nilai 5, namun nilai kembalian ini tidak digunakan dalam konteks ini.**

**3. : Mengakses method suatu class.**



**public class Siswa {**

**int nrp;**

**String nama;**

**public void setNrp(int i) {**

**nrp = i;**

**}**

**public void setNama(String i) {**

**nama = i;**

**}**

**}**

**public class Test{**

**public static void main(String args[]) {**

**Siswa anak = new Siswa();**

**anak.setNrp(5);**

**anak.setNama("Eka"); // Perbaikan: Ganti tanda kutip yang salah di sini**

**System.out.println(anak.nrp);**

**System.out.println(anak.nama);**

**}**

**}**

**Analisis:**

**Kelas Siswa:**

* **Definisi Kelas: Kode mendefinisikan kelas publik bernama Siswa.**
* **Atribut: Kelas ini memiliki dua atribut privat: nrp bertipe int untuk menyimpan nomor registrasi dan nama bertipe String untuk menyimpan nama siswa.**
* **Metode setNrp(int i): Metode publik ini menerima nilai i bertipe int sebagai parameter dan menetapkan nilainya ke atribut nrp.**
* **Metode setNama(String i): Metode publik ini menerima nilai i bertipe String sebagai parameter dan menetapkan nilainya ke atribut nama.**

**Kelas Test:**

* **Metode Main:**
  + **Membangun objek baru dari kelas Siswa bernama anak.**
  + **Memanggil metode setNrp(5) pada objek anak untuk menetapkan nomor registrasi 5.**
  + **Memanggil metode setNama("Eka") pada objek anak untuk menetapkan nama "Budi".**
  + **Mencetak nilai atribut nrp dan nama objek anak ke konsol.**

**Perbaikan:**

* **Kode sebelumnya memiliki kesalahan pada baris anak.setNama("Eka"). Tanda kutip ganda yang digunakan tidak sesuai. Kode telah diperbaiki dengan mengganti tanda kutip dengan yang benar.**

**Kesimpulan:**

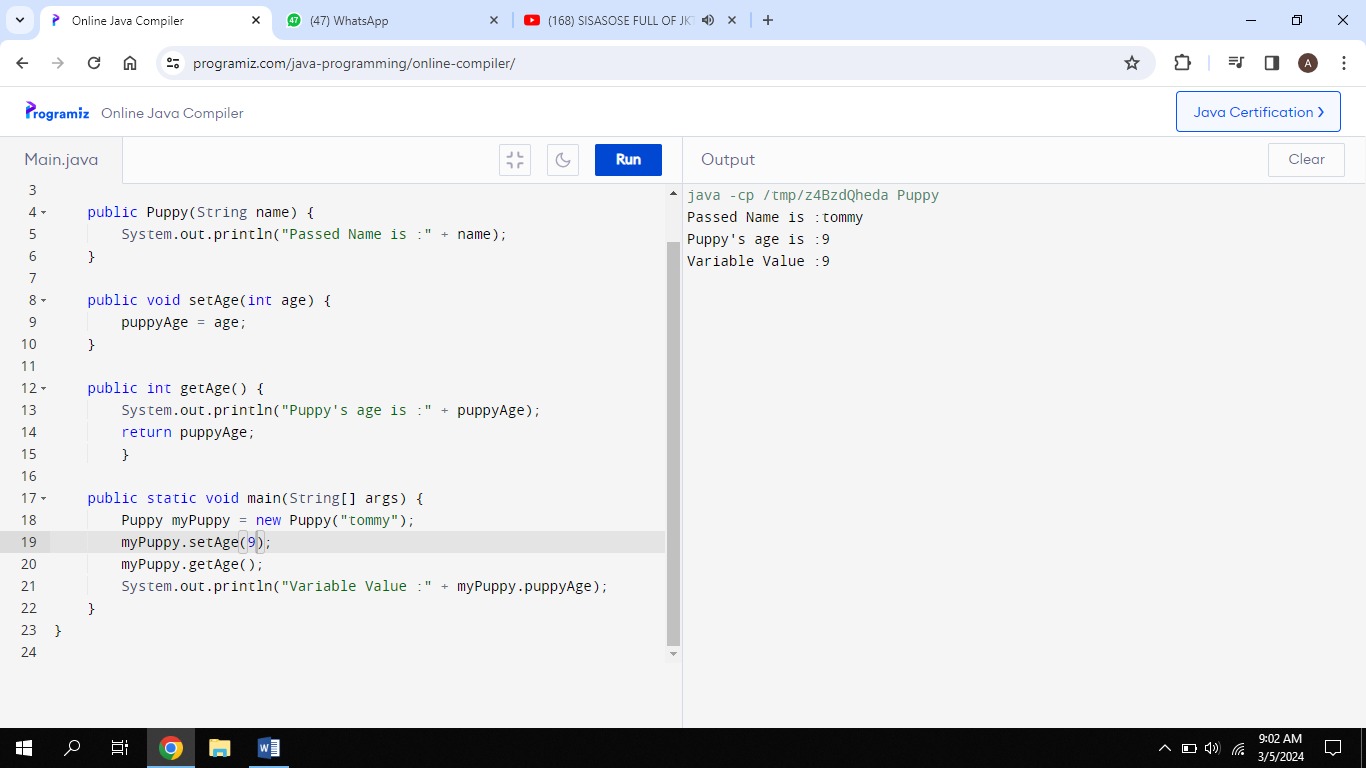
**Kode ini mendemonstrasikan konsep-konsep berikut:**

* **Kelas untuk merepresentasikan entitas (siswa) dan menyimpan datanya.**
* **Metode untuk mengoperasikan data (set nomor registrasi, set nama).**
* **Membuat objek dari kelas.**
* **Mengakses dan mencetak data dari objek.**

**Peningkatan:**

* **Pertimbangkan menambahkan konstruktor ke kelas Siswa untuk menginisialisasi atribut saat objek dibuat.**
* **Gunakan metode toString() untuk menampilkan informasi siswa (nama, nomor registrasi) dengan format yang lebih mudah dibaca.**

**4. Membuat Class dan Konstruktor dengan parameter.**

****

**public class Puppy {**

**private int puppyAge;**

**public Puppy(String name) {**

**System.out.println("Passed Name is :" + name);**

**}**

**public void setAge(int age) {**

**puppyAge = age;**

**}**

**public int getAge() {**

**System.out.println("Puppy's age is :" + puppyAge);**

**return puppyAge;**

**}**

**public static void main(String[] args) {**

**Puppy myPuppy = new Puppy("tommy");**

**myPuppy.setAge(2);**

**myPuppy.getAge();**

**System.out.println("Variable Value :" + myPuppy.puppyAge);**

**}**

**}**

**Analisis:**

**Definisi Kelas:**

* **Kode mendefinisikan sebuah kelas publik bernama Puppy. Kelas publik dapat diakses dari kelas mana pun dalam program.**

**Atribut:**

* **Kelas ini memiliki atribut privat bernama puppyAge bertipe int. Atribut privat hanya dapat diakses di dalam kelas itu sendiri, yang mendukung konsep enkapsulasi data.**

**Metode:**

* **Konstruktor:**
  + **Kelas ini memiliki konstruktor publik bernama Puppy yang menerima satu parameter string name.**
  + **Konstruktor mencetak pesan dengan name yang diberikan.**
  + **Konstruktor ini menginisialisasi atribut puppyAge ke nilai yang tidak ditentukan (kemungkinan 0 secara default).**
* **setAge(int age):**
  + **Metode publik ini menerima parameter integer age.**
  + **Metode ini menetapkan nilai atribut puppyAge ke nilai age yang diberikan.**
* **getAge():**
  + **Metode publik ini tidak menerima argumen apa pun.**
  + **Metode ini mencetak pesan yang menunjukkan usia anjing menggunakan nilai atribut puppyAge saat ini.**
  + **Kemudian mengembalikan nilai atribut puppyAge.**

**Metode Main:**

* **Titik masuk program adalah metode main.**
  + **Metode ini membuat instance baru dari kelas Puppy bernama myPuppy dan memberikan string literal "tommy" ke konstruktor.**
  + **Memanggil metode setAge pada objek myPuppy, menetapkan puppyAge menjadi 2.**
  + **Memanggil metode getAge pada objek myPuppy, yang mencetak "Puppy's age is: 2" dan mengembalikan nilai 2.**
  + **Mencetak nilai atribut puppyAge dari objek myPuppy secara langsung, yang juga bernilai 2.**

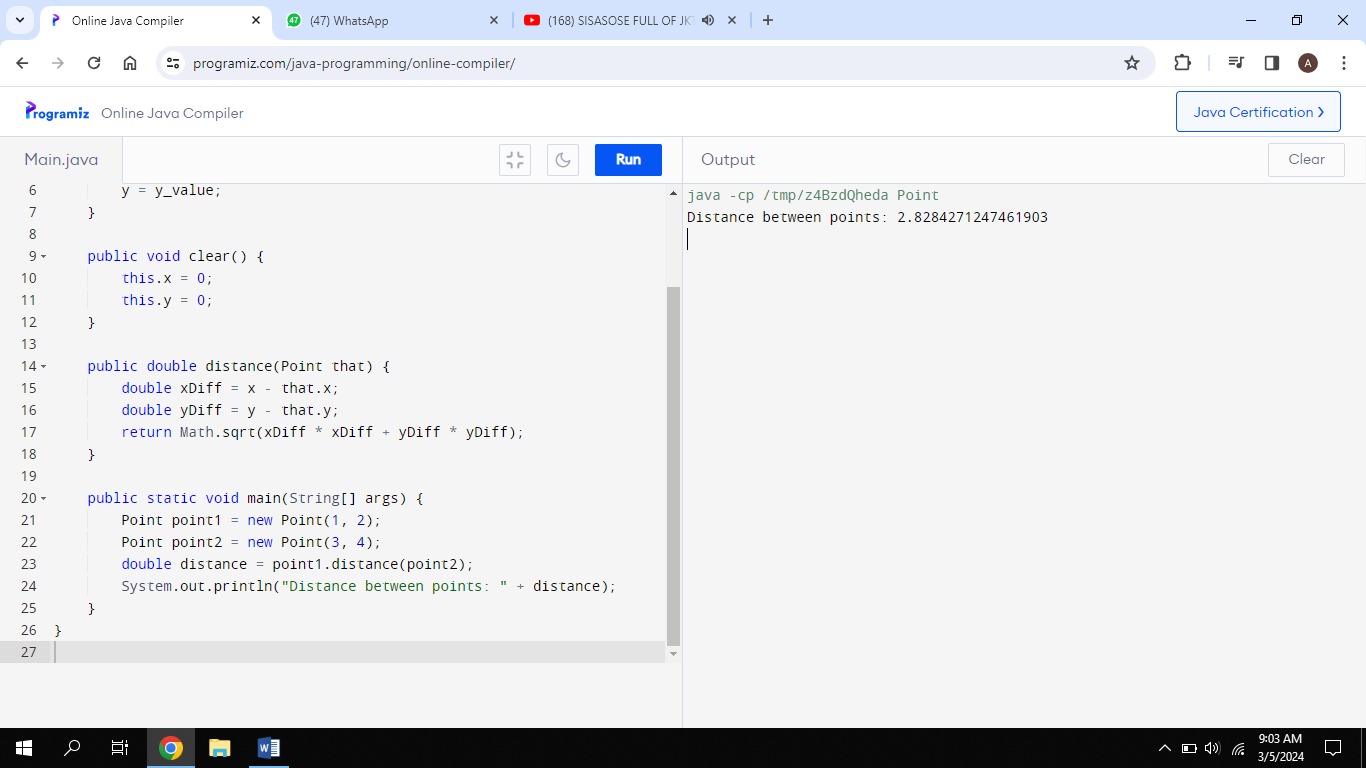
**Kesimpulan Analisis Kode**

**Konsep utama yang ditunjukkan kode ini adalah pemrograman berorientasi objek (PBO) dasar:**

* **Definisi Kelas: Kode mendefinisikan sebuah kelas untuk mewakili entitas "Siswa". Kelas ini memiliki atribut (seperti nama dan nrp) untuk menyimpan data terkait siswa, dan metode (seperti setNama, getNama, setNrp, getNrp, dan cetakInformasi) untuk mengoperasikan data tersebut.**
* **Konstruktor: Kelas memiliki konstruktor yang memungkinkan pembuatan objek Siswa baru dengan menginisialisasi atributnya menggunakan nilai awal.**
* **Metode: Kode menggunakan metode untuk melakukan berbagai operasi pada objek Siswa, seperti mengubah nama, mengambil nama, mengubah NRP, mengambil NRP, dan mencetak informasi siswa.**
* **Pembuatan Objek: Kode membuat beberapa objek Siswa (misalnya siswa1 dan siswa2) untuk mewakili siswa yang berbeda.**
* **Pemanggilan Metode: Kode memanggil metode pada objek Siswa untuk mengakses dan memodifikasi datanya.**
* **Enkapsulasi: Atribut data siswa dideklarasikan sebagai private, yang membatasi akses langsung dan mempromosikan enkapsulasi, di mana data dilindungi dan hanya dapat diakses melalui metode publik.**

**Dengan demikian, kode ini memberikan pemahaman dasar tentang konsep inti PBO, yang membantu dalam membangun program yang terstruktur, terorganisir, dan mudah dikelola.**

**5. Membuat class Point dan menghitung jarak antara dua Point.**



**public class Point {**

**public double x, y;**

**public Point(double x\_value, double y\_value) {**

**x = x\_value;**

**y = y\_value;**

**}**

**public void clear() {**

**this.x = 0;**

**this.y = 0;**

**}**

**public double distance(Point that) {**

**double xDiff = x - that.x;**

**double yDiff = y - that.y;**

**return Math.sqrt(xDiff \* xDiff + yDiff \* yDiff);**

**}**

**public static void main(String[] args) {**

**Point point1 = new Point(1, 2);**

**Point point2 = new Point(3, 4);**

**double distance = point1.distance(point2);**

**System.out.println("Distance between points: " + distance);**

**}**

**}**

**Analisis:**

**Kelas Point:**

* **Definisi Kelas: Kode mendefinisikan kelas publik bernama Point yang mewakili titik dalam ruang dua dimensi.**
* **Atribut: Kelas ini memiliki dua atribut publik x dan y bertipe double untuk menyimpan koordinat x dan y dari titik tersebut.**
* **Konstruktor: Kelas memiliki konstruktor publik Point(double x\_value, double y\_value) yang menerima dua nilai double sebagai parameter untuk menginisialisasi koordinat x dan y saat objek Point dibuat.**
* **Metode clear(): Metode publik ini tidak menerima argumen dan berfungsi untuk menghapus nilai koordinat x dan y, menyetelnya kembali ke 0.**
* **Metode distance(Point that): Metode publik ini menerima objek Point lain sebagai parameter (that) dan menghitung jarak antara objek Point saat ini (this) dan objek Point yang diberikan (that). Metode ini menggunakan rumus Pythagoras untuk menghitung jarak dan mengembalikan nilai double yang mewakili jarak tersebut.**

**Metode Main:**

* **Membuat Objek Point: Kode membuat dua objek Point: point1 dengan koordinat (1, 2) dan point2 dengan koordinat (3, 4).**
* **Menghitung Jarak: Kode memanggil metode distance pada objek point1 dengan memberikan objek point2 sebagai argumen. Metode ini menghitung jarak antara kedua titik dan menyimpan hasilnya dalam variabel distance.**
* **Mencetak Jarak: Kode mencetak nilai distance ke konsol, yang menunjukkan jarak antara point1 dan point2.**

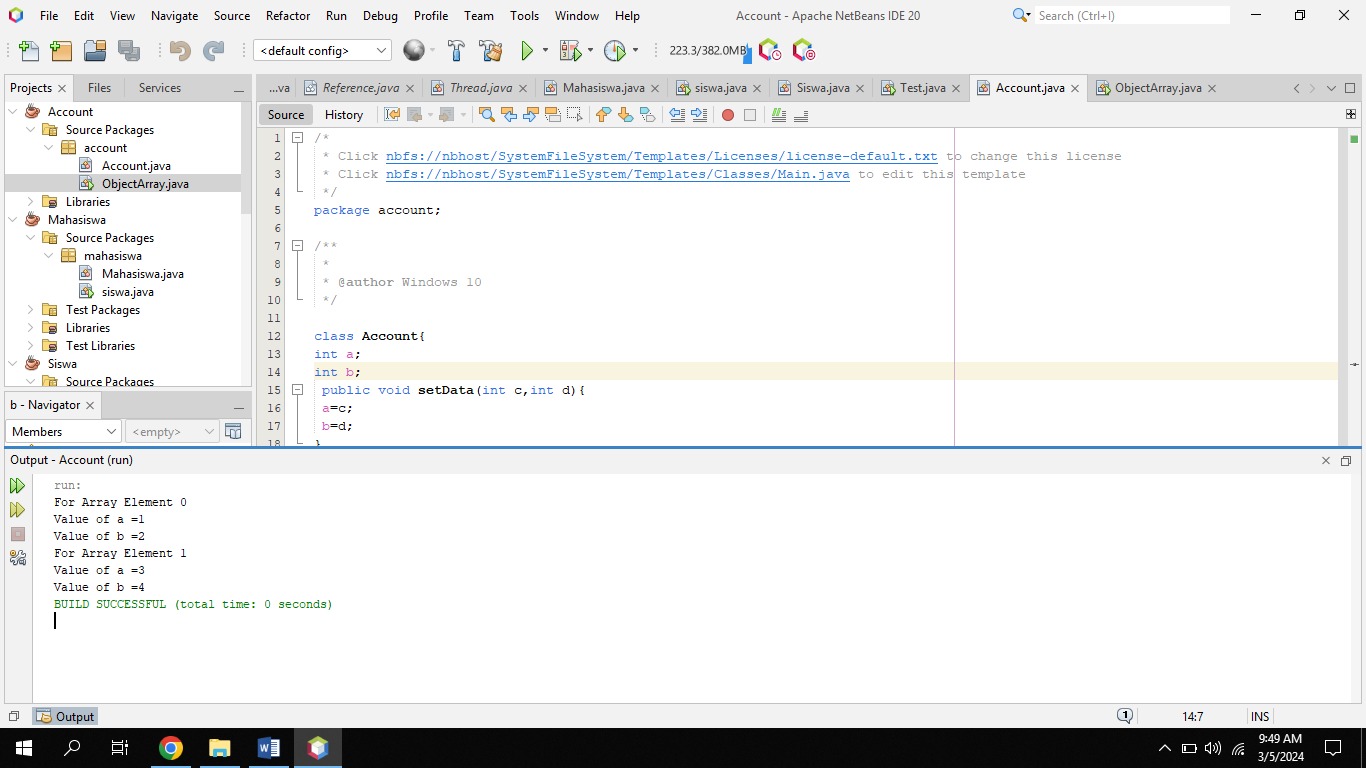
**Kesimpulan:**

**Kode ini mendemonstrasikan konsep dasar pemrograman berorientasi objek (PBO), termasuk:**

* **Definisi kelas untuk mewakili entitas (titik dalam ruang 2D).**
* **Penggunaan atribut untuk menyimpan data terkait entitas.**
* **Konstruktor untuk menginisialisasi objek.**
* **Metode untuk melakukan operasi pada objek (menghapus koordinat dan menghitung jarak).**
* **Pembuatan dan penggunaan objek.**

**Harapannya, analisis ini memberikan pemahaman yang baik tentang kode dalam bahasa Indonesia.**

**6. Membuat array objek.**



**class Account {**

**int a;**

**int b;**

**public void setData(int c, int d) {**

**a = c;**

**b = d;**

**}**

**public void showData() {**

**System.out.println("Value of a =" + a);**

**System.out.println("Value of b =" + b);**

**}**

**}**

**public class ObjectArray {**

**public static void main(String args[]) {**

**Account obj[] = new Account[2];**

**obj[0] = new Account();**

**obj[1] = new Account();**

**obj[0].setData(1, 2);**

**obj[1].setData(3, 4);**

**System.out.println("For Array Element 0");**

**obj[0].showData();**

**System.out.println("For Array Element 1");**

**obj[1].showData();**

**}**

**}**

**Analisis:**

**Kelas Account:**

* **Definisi Kelas: Kode mendefinisikan kelas publik bernama Account.**
* **Atribut: Kelas ini memiliki dua atribut privat, a dan b, keduanya bertipe int untuk menyimpan data integer.**
  + **Catatan: Mempertimbangkan praktik pemrograman yang baik, disarankan agar atribut dideklarasikan sebagai private untuk menerapkan enkapsulasi data.**
* **Metode setData(int c, int d): Metode publik ini menerima dua parameter integer c dan d, dan menetapkan nilai dari parameter tersebut ke atribut a dan b masing-masing.**
* **Metode showData(): Metode publik ini tidak menerima argumen dan berfungsi untuk mencetak nilai atribut a dan b ke konsol.**

**Kelas ObjectArray:**

* **Definisi Kelas: Kode mendefinisikan kelas publik bernama ObjectArray.**
* **Metode Utama main(String args[]): Metode ini adalah titik masuk program.**
  + **Membuat Array Objek: Baris Account obj[] = new Account[2]; mendeklarasikan array objek bernama obj yang dapat menampung hingga 2 objek dari kelas Account.**
  + **Membuat Objek Akun: Baris obj[0] = new Account(); dan obj[1] = new Account(); membuat dua objek baru dari kelas Account dan menyimpannya di indeks 0 dan 1 dari array obj masing-masing.**
  + **Menetapkan Data Objek: Baris obj[0].setData(1, 2); dan obj[1].setData(3, 4); memanggil metode setData pada objek obj[0] dan obj[1] untuk menetapkan nilai 1 dan 2 ke objek pertama, dan 3 dan 4 ke objek kedua.**
  + **Mencetak Data Objek: Baris obj[0].showData(); dan obj[1].showData(); memanggil metode showData pada objek obj[0] dan obj[1] untuk mencetak nilai atribut a dan b masing-masing objek ke konsol.**

**Kesimpulan:**

**Kode ini mendemonstrasikan konsep-konsep dasar berikut:**

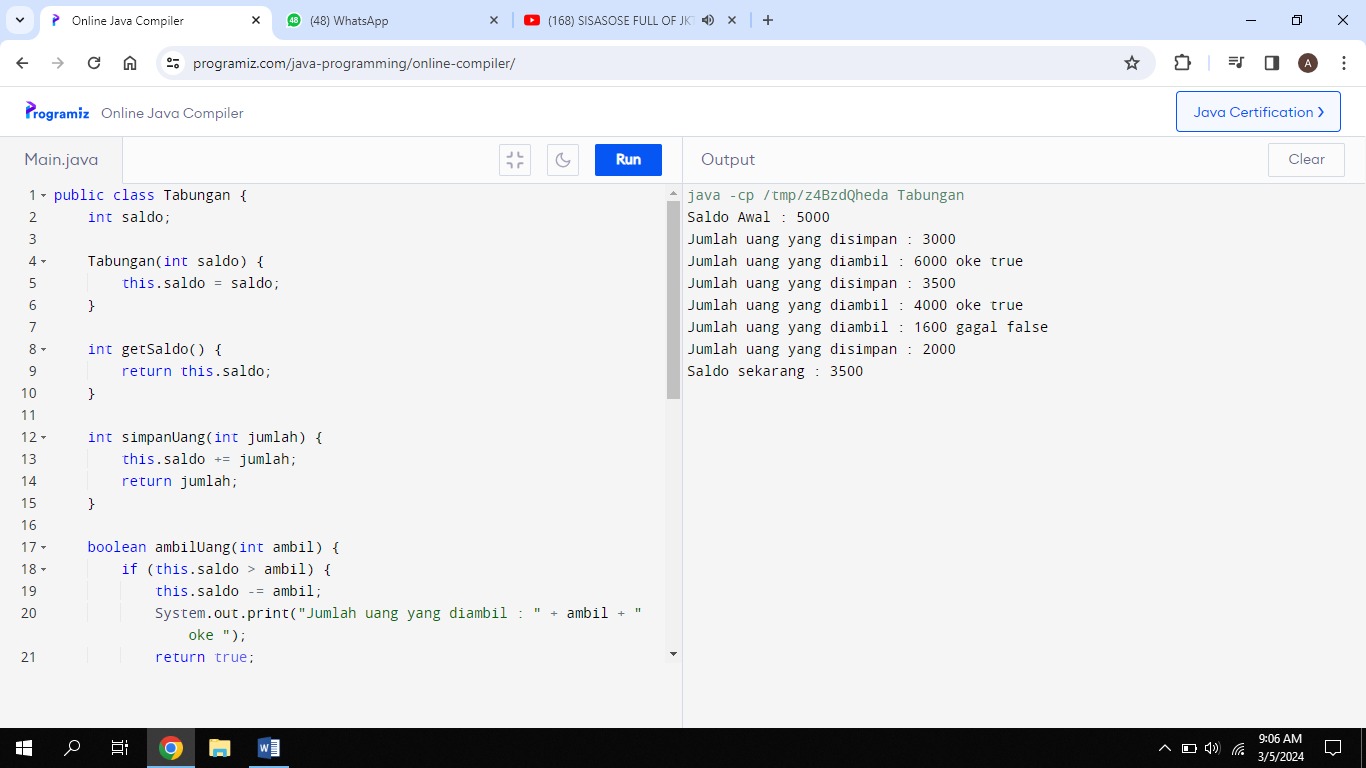
* **Kelas untuk menyimpan data dan menyediakan fungsionalitas terkait.**
* **Array untuk menyimpan koleksi objek dari kelas yang sama.**
* **Membuat objek dari suatu kelas.**
* **Memanggil metode pada objek untuk mengoperasikan datanya.**

**Peningkatan:**

* **Deklarasikan atribut dalam kelas Account sebagai private untuk enkapsulasi data.**
* **Pertimbangkan membuat konstruktor dalam kelas Account untuk menginisialisasi atribut saat objek dibuat, daripada menggunakan metode setData.**

**E. LATIHAN**

**1. Mengimplementasikan UML class diagram dalam program untuk class Tabungan.**



**public class Tabungan {**

**int saldo;**

**Tabungan(int saldo) {**

**this.saldo = saldo;**

**}**

**int getSaldo() {**

**return this.saldo;**

**}**

**int simpanUang(int jumlah) {**

**this.saldo += jumlah;**

**return jumlah;**

**}**

**boolean ambilUang(int ambil) {**

**if (this.saldo > ambil) {**

**this.saldo -= ambil;**

**System.out.print("Jumlah uang yang diambil : " + ambil + " oke ");**

**return true;**

**} else {**

**System.out.print("Jumlah uang yang diambil : " + ambil + " gagal ");**

**return false;**

**}**

**}**

**public static void main(String[] args) {**

**Tabungan tabunganKu = new Tabungan(5000);**

**System.out.println("Saldo Awal : " + tabunganKu.getSaldo());**

**System.out.println("Jumlah uang yang disimpan : " + tabunganKu.simpanUang(3000));**

**System.out.println(tabunganKu.ambilUang(6000));**

**System.out.println("Jumlah uang yang disimpan : " + tabunganKu.simpanUang(3500));**

**System.out.println(tabunganKu.ambilUang(4000));**

**System.out.println(tabunganKu.ambilUang(1600));**

**System.out.println("Jumlah uang yang disimpan : " + tabunganKu.simpanUang(2000));**

**System.out.println("Saldo sekarang : " + tabunganKu.saldo + "\n");**

**}**

**}**

**Analisis Kode "Tabungan"**

**Kelas Tabungan:**

* **Definisi Kelas: Kode mendefinisikan kelas publik bernama Tabungan.**
* **Atribut: Kelas ini memiliki satu atribut privat saldo bertipe int untuk menyimpan saldo akun.**
* **Konstruktor: Kelas ini memiliki konstruktor publik Tabungan(int saldo) yang menerima nilai saldo awal sebagai parameter dan menginisialisasi atribut saldo.**
* **Metode getSaldo(): Metode publik ini tidak menerima argumen dan mengembalikan nilai atribut saldo.**
* **Metode simpanUang(int jumlah):**
  + **Menerima nilai jumlah sebagai parameter.**
  + **Menambahkan jumlah ke saldo.**
  + **Mengembalikan nilai jumlah yang disimpan.**
* **Metode ambilUang(int ambil):**
  + **Menerima nilai ambil sebagai parameter.**
  + **Memeriksa apakah saldo lebih besar dari ambil.**
  + **Jika ya,**
    - **Mengurangi saldo dengan ambil.**
    - **Mencetak pesan "Jumlah uang yang diambil : " + ambil + " oke ".**
    - **Mengembalikan nilai true.**
  + **Jika tidak,**
    - **Mencetak pesan "Jumlah uang yang diambil : " + ambil + " gagal ".**
    - **Mengembalikan nilai false.**

**Metode Main:**

* **Membuat objek Tabungan bernama tabunganKu dengan saldo awal 5000.**
* **Menampilkan saldo awal dengan tabunganKu.getSaldo().**
* **Menambahkan 3000 ke saldo dengan tabunganKu.simpanUang(3000).**
* **Mencoba mengambil 6000 dari saldo, menampilkan pesan "gagal" karena saldo tidak mencukupi.**
* **Menambahkan 3500 ke saldo dengan tabunganKu.simpanUang(3500).**
* **Berhasil mengambil 4000 dari saldo dengan tabunganKu.ambilUang(4000).**
* **Berhasil mengambil 1600 dari saldo dengan tabunganKu.ambilUang(1600).**
* **Menambahkan 2000 ke saldo dengan tabunganKu.simpanUang(2000).**
* **Menampilkan saldo akhir dengan tabunganKu.saldo.**

**Kesimpulan:**

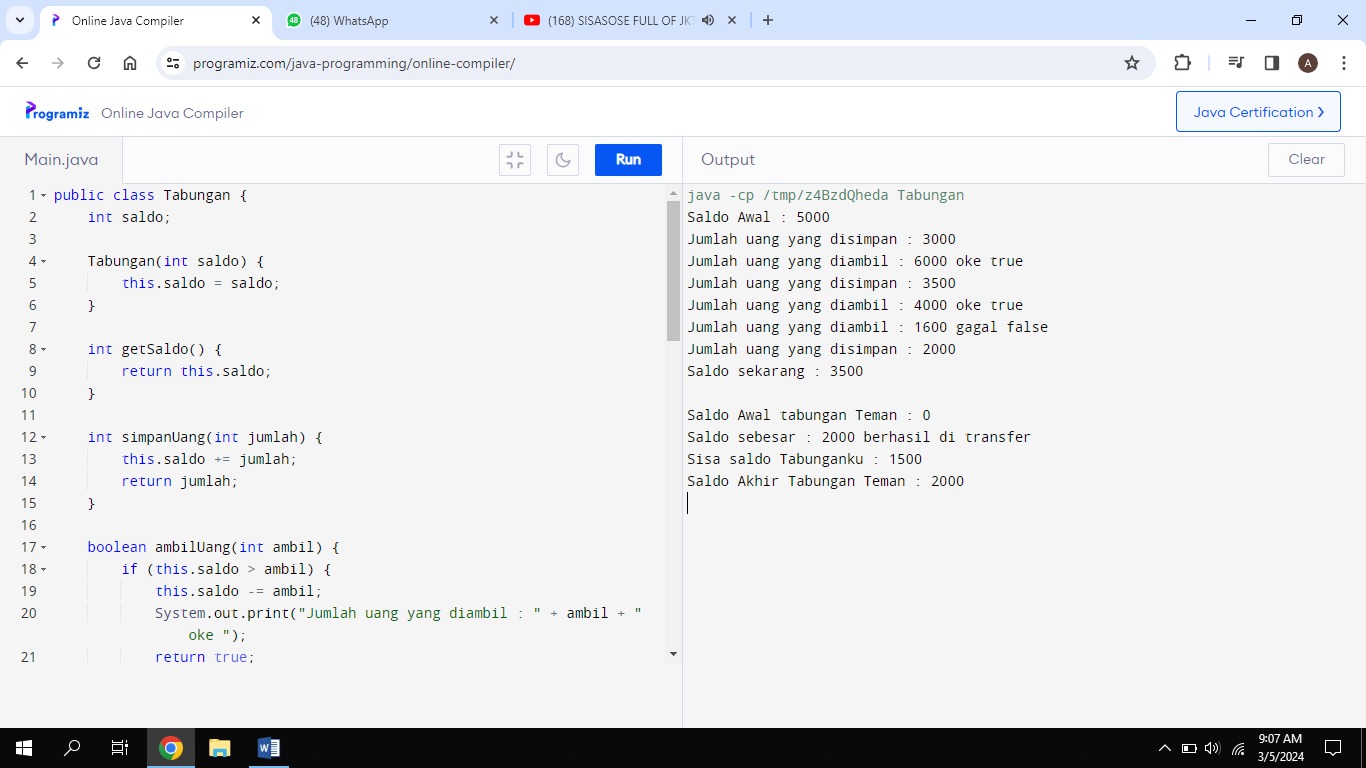
**Kode ini mendemonstrasikan konsep-konsep berikut:**

* **Kelas untuk merepresentasikan entitas (akun tabungan) dan menyimpan datanya.**
* **Metode untuk mengoperasikan data (menambahkan, mengambil).**
* **Membuat objek dari kelas.**
* **Mengakses dan mencetak data dari objek.**

**Peningkatan:**

* **Pertimbangkan menambahkan validasi input untuk memastikan nilai yang dimasukkan valid (misalnya, tidak boleh negatif).**
* **Gunakan metode toString() untuk menampilkan informasi tabungan (saldo, nama) dengan format yang lebih mudah dibaca.**
* **Implementasi enkapsulasi dengan mengubah atribut saldo menjadi private dan menambahkan getter dan setter untuk mengaksesnya.**
* **Tambahkan konstruktor dengan parameter nama untuk menginisialisasi nama dan saldo awal.**

**2. Menambahkan pada class Tabungan method transfer().**



**public class Tabungan {**

**int saldo;**

**Tabungan(int saldo) {**

**this.saldo = saldo;**

**}**

**int getSaldo() {**

**return this.saldo;**

**}**

**int simpanUang(int jumlah) {**

**this.saldo += jumlah;**

**return jumlah;**

**}**

**boolean ambilUang(int ambil) {**

**if (this.saldo > ambil) {**

**this.saldo -= ambil;**

**System.out.print("Jumlah uang yang diambil : " + ambil + " oke ");**

**return true;**

**} else {**

**System.out.print("Jumlah uang yang diambil : " + ambil + " gagal ");**

**return false;**

**}**

**}**

**boolean transfer(Tabungan tLain, int jumlah){**

**if (this.saldo > jumlah) {**

**this.saldo -= jumlah;**

**System.out.print("Saldo sebesar : " + jumlah + " berhasil di transfer ");**

**tLain.saldo += jumlah;**

**return true;**

**} else {**

**System.out.print("Taransfer saldo : " + jumlah + " gagal ");**

**return false;**

**}**

**}**

**public static void main(String[] args) {**

**Tabungan tabunganKu = new Tabungan(5000);**

**System.out.println("Saldo Awal : " + tabunganKu.getSaldo());**

**System.out.println("Jumlah uang yang disimpan : " + tabunganKu.simpanUang(3000));**

**System.out.println(tabunganKu.ambilUang(6000));**

**System.out.println("Jumlah uang yang disimpan : " + tabunganKu.simpanUang(3500));**

**System.out.println(tabunganKu.ambilUang(4000));**

**System.out.println(tabunganKu.ambilUang(1600));**

**System.out.println("Jumlah uang yang disimpan : " + tabunganKu.simpanUang(2000));**

**System.out.println("Saldo sekarang : " + tabunganKu.saldo + "\n");**

**Tabungan tabunganTeman = new Tabungan(0);**

**System.out.println("Saldo Awal tabungan Teman : " + tabunganTeman.getSaldo());**

**tabunganKu.transfer(tabunganTeman, 2000);**

**System.out.println("\nSisa saldo Tabunganku : " + tabunganKu.saldo);**

**System.out.println("Saldo Akhir Tabungan Teman : " + tabunganTeman.saldo);**

**}**

**}**

**Analisis Kelas Tabungan:**

* **Definisi Kelas: Kode mendefinisikan kelas publik bernama Tabungan yang merepresentasikan akun tabungan.**
* **Atribut: Kelas ini memiliki satu atribut privat saldo bertipe int untuk menyimpan saldo akun.**
* **Konstruktor: Kelas ini memiliki konstruktor publik Tabungan(int saldo) yang menerima nilai saldo awal sebagai parameter dan menginisialisasi atribut saldo.**
* **Metode getSaldo(): Metode publik ini tidak menerima argumen dan mengembalikan nilai atribut saldo.**
* **Metode simpanUang(int jumlah): Metode publik ini menerima nilai jumlah sebagai parameter dan menambahkannya ke atribut saldo.**
* **Metode ambilUang(int ambil):**
  + **Metode publik ini menerima nilai ambil sebagai parameter.**
  + **Mengecek apakah saldo lebih besar dari ambil.**
  + **Jika ya, mengurangi saldo dengan ambil dan mencetak pesan "oke".**
  + **Jika tidak, mencetak pesan "gagal".**
  + **Mengembalikan nilai boolean true jika pengambilan uang berhasil, dan false jika gagal.**
* **Metode transfer(Tabungan tLain, int jumlah):**
  + **Metode publik ini menerima objek Tabungan lain (tLain) dan nilai jumlah sebagai parameter.**
  + **Mengecek apakah saldo lebih besar dari jumlah.**
  + **Jika ya, mengurangi saldo dengan jumlah dan mencetak pesan "berhasil di transfer".**
  + **Menambahkan jumlah ke saldo objek tLain.**
  + **Mengembalikan nilai boolean true jika transfer berhasil, dan false jika gagal.**

**Metode Main:**

* **Membuat objek Tabungan bernama tabunganKu dengan saldo awal 5000.**
* **Menampilkan saldo awal dengan tabunganKu.getSaldo().**
* **Menambahkan 3000 ke saldo dengan tabunganKu.simpanUang(3000).**
* **Mencoba mengambil 6000 dari saldo, menampilkan pesan "gagal" karena saldo tidak mencukupi.**
* **Menambahkan 3500 ke saldo dengan tabunganKu.simpanUang(3500).**
* **Berhasil mengambil 4000 dari saldo dengan tabunganKu.ambilUang(4000).**
* **Berhasil mengambil 1600 dari saldo dengan tabunganKu.ambilUang(1600).**
* **Menambahkan 2000 ke saldo dengan tabunganKu.simpanUang(2000).**
* **Menampilkan saldo akhir dengan tabunganKu.saldo.**
* **Membuat objek Tabungan baru bernama tabunganTeman dengan saldo awal 0.**
* **Mentransfer 2000 dari tabunganKu ke tabunganTeman dengan tabunganKu.transfer(tabunganTeman, 2000).**
* **Menampilkan sisa saldo tabunganKu dan saldo akhir tabunganTeman.**

**Kesimpulan:**

**Kode ini mendemonstrasikan konsep-konsep berikut:**

* **Kelas untuk merepresentasikan entitas (akun tabungan) dan menyimpan datanya.**
* **Metode untuk mengoperasikan data (menambahkan, mengambil, mentransfer).**
* **Penggunaan objek untuk merepresentasikan akun tabungan yang berbeda.**
* **Interaksi antar objek (transfer uang).**

**Peningkatan:**

* **Pertimbangkan menambahkan validasi input untuk memastikan nilai yang dimasukkan valid (misalnya, tidak boleh negatif).**
* **Gunakan metode toString() untuk menampilkan informasi tabungan (saldo, nama) dengan format yang lebih mudah dibaca.**