Nama :Ahmad Reza Asari

Nim :4123019

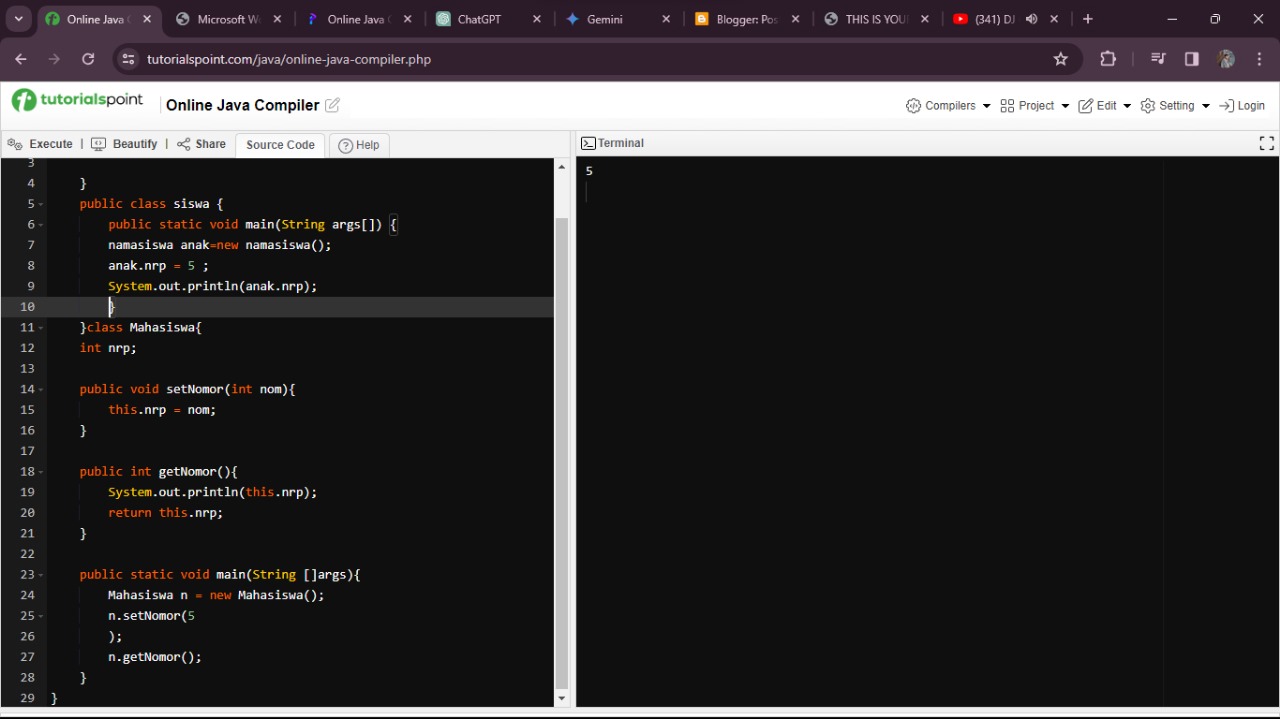
Kelas : A

Semester : 2

Tugas Minggu ke- 2 Class dan Objek

Percobaan

**Nomor 1**



1. class namasiswa {
2. int nrp;
4. }
5. public class siswa {
6. public static void main(String args[]) {
7. namasiswa anak=new namasiswa();
8. anak.nrp = 5 ;
9. System.out.println(anak.nrp);
10. }
11. }

Penjelasan: Program di atas adalah implementasi sederhana dari dua kelas Java, yaitu namasiswa dan siswa. Berikut adalah penjelasan fungsi programnya:

1. \*Deklarasi Kelas namasiswa:\*

- Kelas ini memiliki satu atribut (int nrp), yang dapat menyimpan nomor registrasi siswa.

2. \*Kelas siswa:\*

- Kelas utama yang berisi metode main.

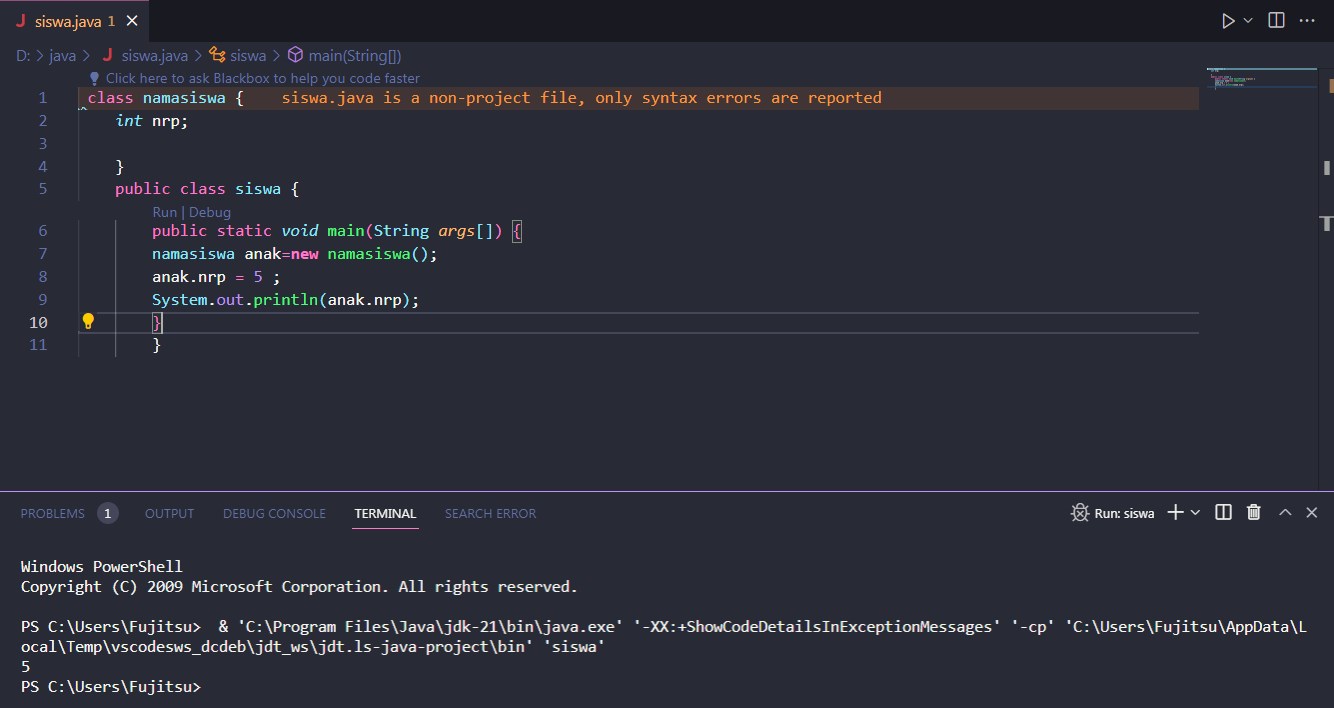
- Membuat objek baru dari kelas namasiswa dengan nama anak.

- Mengatur nilai atribut nrp pada objek anak menjadi 5 dengan menggunakan anak.nrp = 5.

- Mencetak nilai atribut nrp dari objek anak ke konsol menggunakan System.out.println.

Jadi, program ini menciptakan objek dari kelas namasiswa, mengatur nilai atribut nrp, dan kemudian mencetak nilai tersebut ke konsol. Pada contoh ini, nilai yang dicetak adalah 5.

**Nomor 2**



1. class Mahasiswa{
2. int nrp;
3. public void setNomor(int nom){
4. this.nrp = nom;
5. }
6. public int getNomor(){
7. System.out.println(this.nrp);
8. return this.nrp;
9. }
10. public static void main(String []args){
11. Mahasiswa n = new Mahasiswa();
12. n.setNomor(5
13. );
14. n.getNomor();
15. }
16. }

Penjelasan: Program di atas adalah implementasi sederhana dari sebuah kelas Java bernama Mahasiswa. Berikut adalah penjelasan fungsi programnya:

1. \*Deklarasi Kelas Mahasiswa:\*

- Kelas Mahasiswa memiliki satu atribut (int nrp) yang merupakan nomor registrasi mahasiswa.

2. \*Metode setNomor(int nom):\*

- Metode ini digunakan untuk mengatur nilai atribut nrp dengan nilai yang diberikan sebagai argumen.

3. \*Metode getNomor():\*

- Metode ini mencetak nilai nrp ke konsol menggunakan System.out.println dan mengembalikan nilai nrp.

4. \*Metode main(String[] args):\*

- Metode utama program yang membuat objek Mahasiswa baru (n), mengatur nomor registrasi objek tersebut menjadi 5 dengan menggunakan setNomor(5), dan kemudian memanggil getNomor() untuk mencetak dan mengambil nilai nomor registrasi.

Jadi, program ini menciptakan objek mahasiswa, mengatur nomor registrasinya, dan kemudian mencetak nomor registrasi tersebut ke konsol. Pada contoh ini, nomor registrasi yang dicetak adalah 5

**Nomor 3**



1. public class Siswa {
2. int nrp;
3. String nama;
4. public void setNrp(int i) {
5. nrp = i;
6. }
7. public void setNama(String i) {
8. nama = i;
9. }
10. }
11. public class Test {
12. public static void main(String args[]) {
13. Siswa anak = new Siswa();
14. anak.setNrp(5);
15. anak.setNama("Budi"); // Perbaikan: Ganti tanda kutip yang salah di sini
16. System.out.println(anak.nrp);
17. System.out.println(anak.nama);
18. }
19. }

Penjelasan: fungsi dari kodingan di atas:

1. \*Kelas Siswa:\*

- int nrp: Variabel untuk menyimpan nomor registrasi mahasiswa.

- String nama: Variabel untuk menyimpan nama mahasiswa.

- setNrp(int i): Metode untuk mengatur nilai nrp.

- setNama(String i): Metode untuk mengatur nilai nama.

2. \*Kelas Test:\*

- main(String args[]): Metode utama sebagai titik masuk eksekusi program.

- Siswa anak = new Siswa(): Membuat objek dari kelas Siswa dengan nama anak.

- anak.setNrp(5): Menetapkan nilai nrp objek anak menjadi 5.

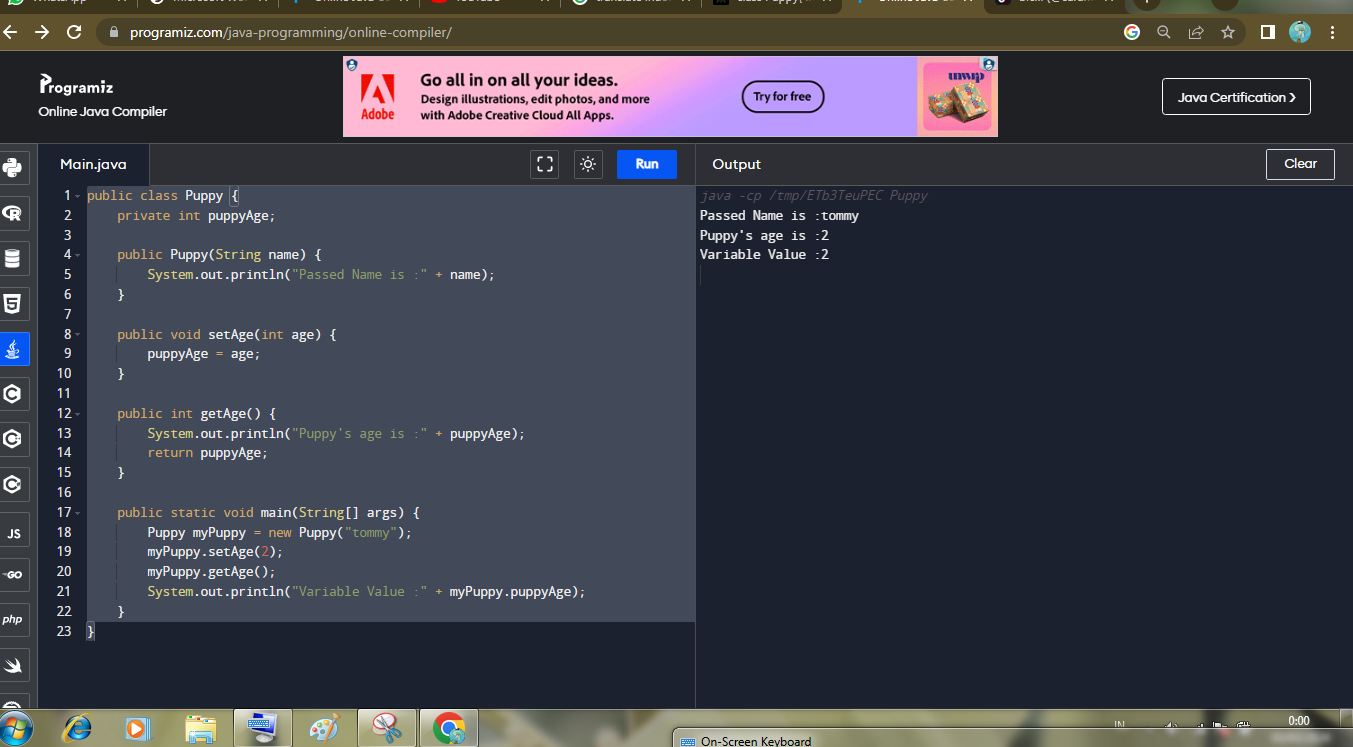
- anak.setNama("Budi"): Menetapkan nilai nama objek anak menjadi "Budi".

- System.out.println(anak.nrp): Mencetak nilai nrp dari objek anak.

- System.out.println(anak.nama): Mencetak nilai nama dari objek anak.

Jadi, program ini menciptakan objek mahasiswa, mengatur nilai nrp dan nama melalui metode setter, dan kemudian mencetak nilai nrp dan nama tersebut. Dengan menggunakan kelas dan objek, ini menunjukkan konsep dasar pemrograman berorientasi objek di Java

**Nomor 4**



1. public class Puppy {
2. private int puppyAge;
3. public Puppy(String name) {
4. System.out.println("Passed Name is :" + name);
5. }
6. public void setAge(int age) {
7. puppyAge = age;
8. }
9. public int getAge() {
10. System.out.println("Puppy's age is :" + puppyAge);
11. return puppyAge;
12. }
13. public static void main(String[] args) {
14. Puppy myPuppy = new Puppy("tommy");
15. myPuppy.setAge(2);
16. myPuppy.getAge();
17. System.out.println("Variable Value :" + myPuppy.puppyAge);
18. }
19. }

Penjelasan: Program di atas adalah implementasi kelas Puppy dalam bahasa pemrograman Java, yang memiliki atribut puppyAge dan beberapa metode untuk mengatur dan mendapatkan nilai usia seekor anak anjing. Berikut adalah penjelasan fungsi programnya:

1. \*Deklarasi Kelas Puppy:\*

- Kelas ini memiliki atribut privat (private int puppyAge) untuk menyimpan usia anak anjing.

2. \*Konstruktor Puppy(String name):\*

- Konstruktor menerima parameter nama dan mencetak pesan yang menyatakan nama yang dilewatkan.

3. \*Metode setAge(int age):\*

- Metode ini digunakan untuk mengatur nilai atribut puppyAge dengan nilai usia yang diberikan sebagai argumen.

4. \*Metode getAge():\*

- Metode ini mencetak usia anjing ke konsol dan mengembalikan nilai usia tersebut.

5. \*Metode main(String[] args):\*

- Metode utama program yang membuat objek myPuppy dari kelas Puppy dengan nama "tommy".

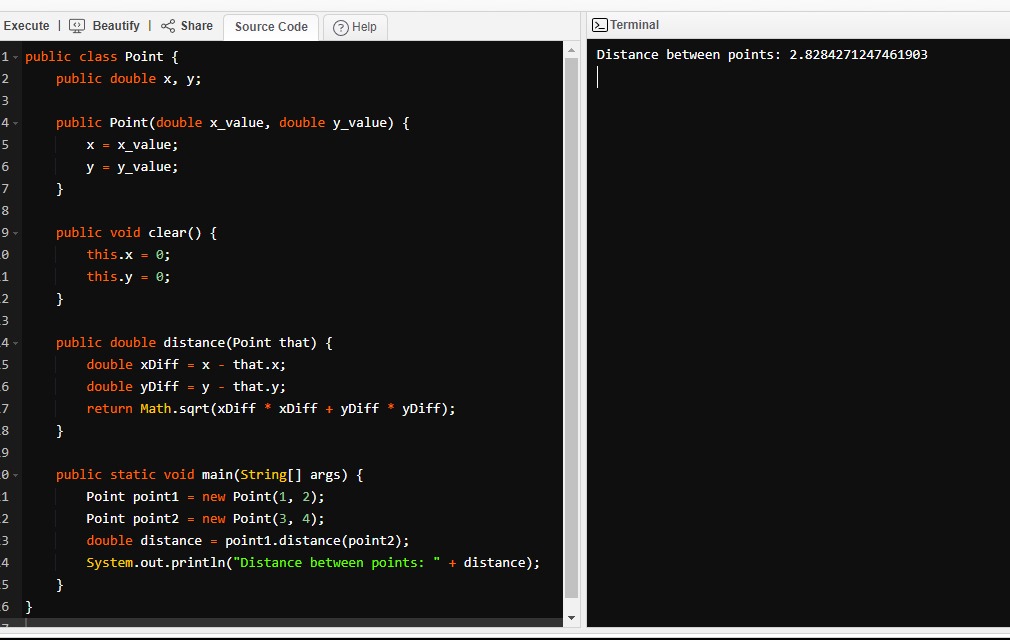
- Mengatur usia anak anjing menggunakan metode setAge(2).

- Mendapatkan dan mencetak usia anak anjing ke konsol dengan metode getAge().

- Mencetak nilai atribut puppyAge secara langsung ke konsol.

Jadi, program ini menciptakan objek Puppy, mengatur dan mendapatkan nilai usia anak anjing, dan mencetak informasi terkait ke konsol. Pada contoh ini, nama anjing dicetak saat objek dibuat, usia diatur dan dicetak, serta nilai atribut usia diakses dan dicetak langsung.

**Nomor 5**



1. public class Point {
2. public double x, y;
4. public Point(double x\_value, double y\_value) {
5. x = x\_value;
6. y = y\_value;
7. }
8. public void clear() {
9. this.x = 0;
10. this.y = 0;
11. }
12. public double distance(Point that) {
13. double xDiff = x - that.x;
14. double yDiff = y - that.y;
15. return Math.sqrt(xDiff \* xDiff + yDiff \* yDiff);
16. }
17. public static void main(String[] args) {
18. Point point1 = new Point(1, 2);
19. Point point2 = new Point(3, 4);
20. double distance = point1.distance(point2);
21. System.out.println("Distance between points: " + distance);
22. }
23. }

Penjelasan : Pertama, dua objek Point dibuat dengan koordinat yang berbeda.

Point point1 = new Point(1, 2); menciptakan objek Point pertama dengan koordinat (1, 2).

Point point2 = new Point(3, 4); menciptakan objek Point kedua dengan koordinat (3, 4).

Kemudian, metode distance() dari objek point1 dipanggil dengan argumen point2.

Metode distance() menghitung jarak antara titik point1 dan point2 menggunakan rumus jarak Euclidean.

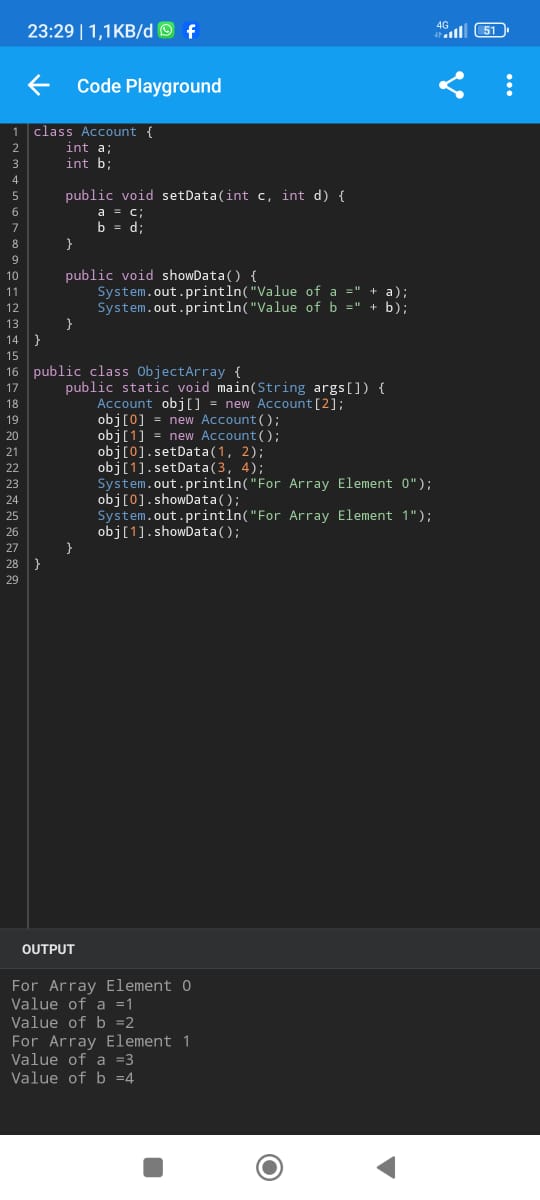
Jarak tersebut kemudian disimpan dalam variabel distance.

Terakhir, hasil jarak yang telah dihitung dicetak ke konsol.

System.out.println("Distance between points: " + distance); mencetak pesan "Distance between points: " diikuti dengan nilai jarak antara dua titik yang telah dihitung sebelumnya.

Jadi, keseluruhan fungsi dari hasil kodingan tersebut adalah untuk menghitung dan mencetak jarak antara dua titik dalam koordinat kartesian.

**Nomor 6**



1. class Account {
2. int a;
3. int b;
4. public void setData(int c, int d) {
5. a = c;
6. b = d;
7. }
8. public void showData() {
9. System.out.println("Value of a =" + a);
10. System.out.println("Value of b =" + b);
11. }
12. }
13. public class ObjectArray {
14. public static void main(String args[]) {
15. Account obj[] = new Account[2];
16. obj[0] = new Account();
17. obj[1] = new Account();
18. obj[0].setData(1, 2);
19. obj[1].setData(3, 4);
20. System.out.println("For Array Element 0");
21. obj[0].showData();
22. System.out.println("For Array Element 1");
23. obj[1].showData();
24. }
25. }

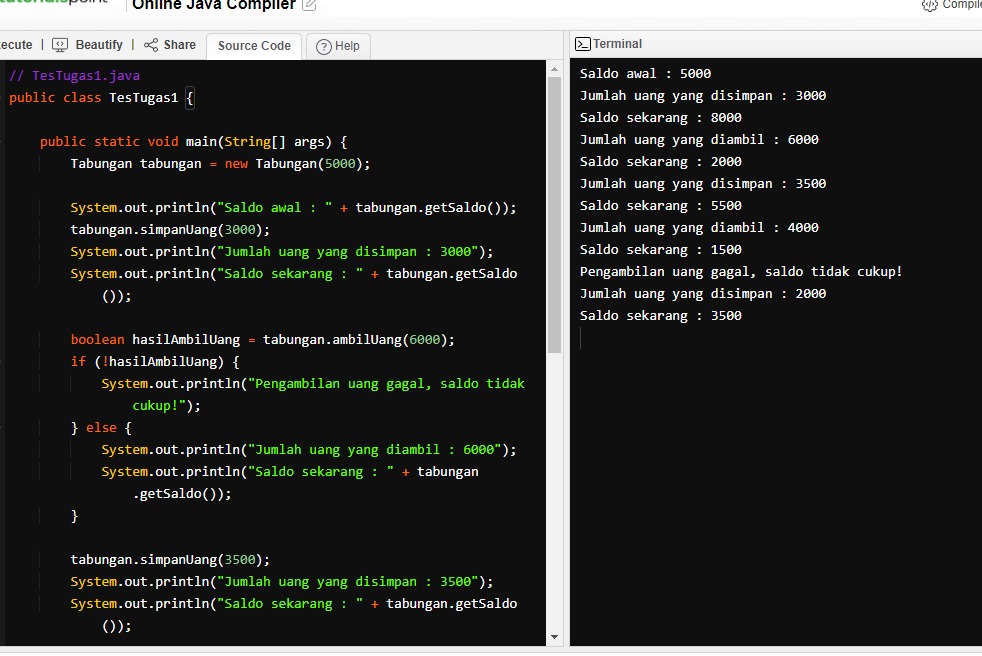
Penjelasan : Kode di atas mendefinisikan dua kelas: Account dan ObjectArray.

Kelas Account memiliki dua variabel anggota a dan b, serta dua metode: setData(int c, int d) untuk mengatur nilai a dan b, dan showData() untuk menampilkan nilai a dan b.

Kelas ObjectArray adalah kelas utama yang memiliki metode main(String args[]). Di dalam metode main, sebuah array obj dari tipe Account dengan panjang 2 dibuat. Kemudian, dua objek Account dibuat dan disimpan di dalam array tersebut. Nilai a dan b diatur menggunakan metode setData() untuk masing-masing objek. Akhirnya, nilai a dan b untuk masing-masing objek ditampilkan menggunakan metode showData().

PERCOBAAN E

**Nomor 1**



1. public class TesTugas1 {
2. public static void main(String[] args) {
3. Tabungan tabungan = new Tabungan(5000);
4. System.out.println("Saldo awal : " + tabungan.getSaldo());
5. tabungan.simpanUang(3000);
6. System.out.println("Jumlah uang yang disimpan : 3000");
7. System.out.println("Saldo sekarang : " + tabungan.getSaldo());
8. boolean hasilAmbilUang = tabungan.ambilUang(6000);
9. if (!hasilAmbilUang) {
10. System.out.println("Pengambilan uang gagal, saldo tidak cukup!");
11. } else {
12. System.out.println("Jumlah uang yang diambil : 6000");
13. System.out.println("Saldo sekarang : " + tabungan.getSaldo());
14. }
15. tabungan.simpanUang(3500);
16. System.out.println("Jumlah uang yang disimpan : 3500");
17. System.out.println("Saldo sekarang : " + tabungan.getSaldo());
18. hasilAmbilUang = tabungan.ambilUang(4000);
19. if (!hasilAmbilUang) {
20. System.out.println("Pengambilan uang gagal, saldo tidak cukup!");
21. } else {
22. System.out.println("Jumlah uang yang diambil : 4000");
23. System.out.println("Saldo sekarang : " + tabungan.getSaldo());
24. }
25. hasilAmbilUang = tabungan.ambilUang(1600);
26. if (!hasilAmbilUang) {
27. System.out.println("Pengambilan uang gagal, saldo tidak cukup!");
28. } else {
29. System.out.println("Jumlah uang yang diambil : 1600");
30. System.out.println("Saldo sekarang : " + tabungan.getSaldo());
31. }
32. tabungan.simpanUang(2000);
33. System.out.println("Jumlah uang yang disimpan : 2000");
34. System.out.println("Saldo sekarang : " + tabungan.getSaldo());
35. }
36. }

Penjelasan : Program di atas adalah implementasi sederhana dari kelas Tabungan dalam bahasa pemrograman Java. Berikut adalah penjelasan fungsi programnya:

1. \*Deklarasi Kelas Tabungan:\*

- Kelas ini memiliki atribut privat (private int saldo) untuk menyimpan saldo tabungan.

2. \*Konstruktor Tabungan(int initSaldo):\*

- Konstruktor ini digunakan untuk membuat objek Tabungan baru dengan saldo awal yang diberikan sebagai argumen saat objek dibuat.

3. \*Metode getSaldo():\*

- Metode ini mengembalikan nilai saldo tabungan.

4. \*Metode simpanUang(int jumlah):\*

- Metode ini digunakan untuk menambahkan jumlah uang ke saldo tabungan.

5. \*Metode ambilUang(int jumlah):\*

- Metode ini memeriksa apakah saldo mencukupi untuk pengambilan uang sejumlah tertentu.

- Jika mencukupi, saldo dikurangkan dan metode mengembalikan true.

- Jika tidak mencukupi, metode mengembalikan false.

6. \*Metode transfer(Tabungan tab, int jumlah):\*

- Metode ini memeriksa apakah saldo mencukupi untuk transfer sejumlah tertentu.

- Jika mencukupi, saldo objek saat ini dikurangkan dan saldo objek tab ditambahkan.

- Metode mengembalikan true untuk transfer berhasil.

- Jika saldo tidak mencukupi, metode mengembalikan false.

7. \*Metode main(String[] args):\*

- Metode utama program yang membuat dua objek Tabungan (tabungan1 dan tabungan2) dengan saldo awal masing-masing.

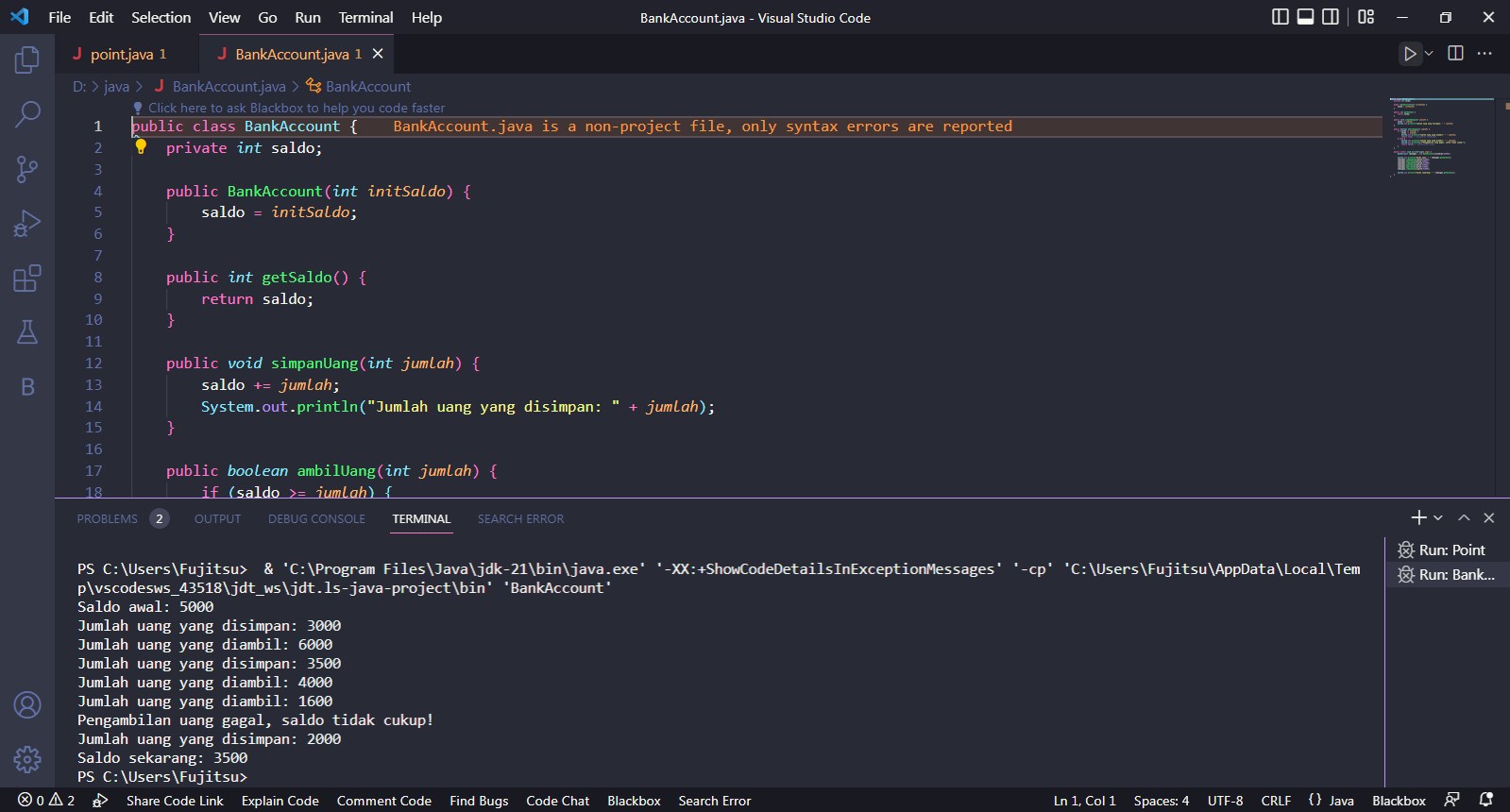
- Menampilkan saldo awal kedua tabungan.

- Menambahkan uang ke tabungan1, menampilkan saldo baru.

- Mencoba transfer dari tabungan1 ke tabungan2, menampilkan hasil dan saldo kedua tabungan setelah transfer.

Jadi, program ini menyimulasikan penggunaan objek Tabungan dengan menyimpan, mengambil uang, dan melakukan transfer saldo antar dua tabungan.

**Nomor 2**



1. public class BankAccount {
2. private int saldo;
3. public BankAccount(int initSaldo) {
4. saldo = initSaldo;
5. }
6. public int getSaldo() {
7. return saldo;
8. }
9. public void simpanUang(int jumlah) {
10. saldo += jumlah;
11. System.out.println("Jumlah uang yang disimpan: " + jumlah);
12. }
13. public boolean ambilUang(int jumlah) {
14. if (saldo >= jumlah) {
15. saldo -= jumlah;
16. System.out.println("Jumlah uang yang diambil: " + jumlah);
17. return true; // Successful withdrawal
18. } else {
19. System.out.println("Jumlah uang yang diambil: " + jumlah);
20. System.out.println("Pengambilan uang gagal, saldo tidak cukup!");
21. return false; // Insufficient funds
22. }
23. }
24. public static void main(String[] args) {
25. BankAccount tabungan = new BankAccount(5000);
26. System.out.println("Saldo awal: " + tabungan.getSaldo());
27. tabungan.simpanUang(3000);
28. tabungan.ambilUang(6000);
29. tabungan.simpanUang(3500);
30. tabungan.ambilUang(4000);
31. tabungan.ambilUang(1600);
32. tabungan.simpanUang(2000);
33. System.out.println("Saldo sekarang: " + tabungan.getSaldo());
34. }
35. }

Penjelasan: Program di atas adalah implementasi sederhana dari kelas BankAccount dalam bahasa pemrograman Java. Berikut adalah penjelasan fungsi programnya:

1. \*Deklarasi Kelas BankAccount:\*

- Kelas ini memiliki atribut privat (private int saldo) untuk menyimpan saldo akun bank.

2. \*Konstruktor BankAccount(int initSaldo):\*

- Konstruktor ini digunakan untuk membuat objek BankAccount baru dengan saldo awal yang diberikan sebagai argumen saat objek dibuat.

3. \*Metode getSaldo():\*

- Metode ini mengembalikan nilai saldo akun bank.

4. \*Metode simpanUang(int jumlah):\*

- Metode ini digunakan untuk menambahkan jumlah uang ke saldo akun bank.

- Mencetak jumlah uang yang disimpan.

5. \*Metode ambilUang(int jumlah):\*

- Metode ini memeriksa apakah saldo mencukupi untuk pengambilan uang sejumlah tertentu.

- Jika mencukupi, saldo dikurangkan, mencetak jumlah uang yang diambil, dan mengembalikan true.

- Jika tidak mencukupi, mencetak jumlah uang yang diambil, pesan kegagalan, dan mengembalikan false.

6. \*Metode main(String[] args):\*

- Metode utama program yang membuat objek BankAccount (tabungan) dengan saldo awal 5000.

- Menampilkan saldo awal.

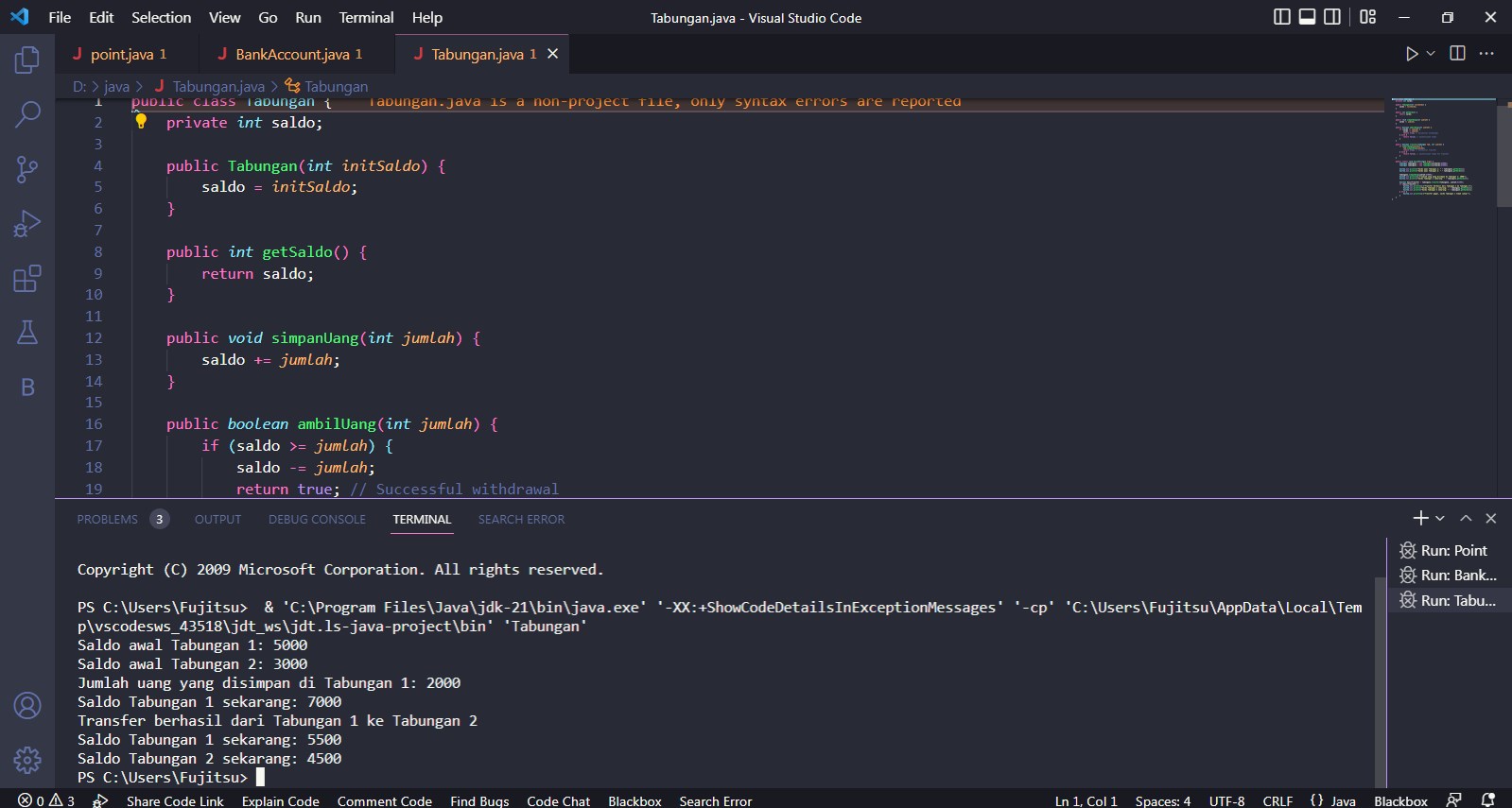
- Menambahkan uang, mencetak jumlah dan saldo baru.

- Mengambil uang beberapa kali, mencetak jumlah dan saldo baru setiap kali.

- Menampilkan saldo akhir setelah operasi-operasi tersebut.

Jadi, program ini menyimulasikan operasi perbankan sederhana dengan membuat objek BankAccount, menyimpan uang, dan mengambil uang, sambil mencetak informasi yang sesuai ke konsol.

**Nomor 3**



1. public class Tabungan {
2. private int saldo;
3. public Tabungan(int initSaldo) {
4. saldo = initSaldo;
5. }
6. public int getSaldo() {
7. return saldo;
8. }
9. public void simpanUang(int jumlah) {
10. saldo += jumlah;
11. }
12. public boolean ambilUang(int jumlah) {
13. if (saldo >= jumlah) {
14. saldo -= jumlah;
15. return true; // Successful withdrawal
16. } else {
17. return false; // Insufficient funds
18. }
19. }
20. public boolean transfer(Tabungan tab, int jumlah) {
21. if (ambilUang(jumlah)) {
22. tab.simpanUang(jumlah);
23. return true; // Successful transfer
24. } else {
25. return false; // Insufficient funds for transfer
26. }
27. }
28. public static void main(String[] args) {
29. Tabungan tabungan1 = new Tabungan(5000);
30. Tabungan tabungan2 = new Tabungan(3000);
31. System.out.println("Saldo awal Tabungan 1: " + tabungan1.getSaldo());
32. System.out.println("Saldo awal Tabungan 2: " + tabungan2.getSaldo());
33. tabungan1.simpanUang(2000);
34. System.out.println("Jumlah uang yang disimpan di Tabungan 1: 2000");
35. System.out.println("Saldo Tabungan 1 sekarang: " + tabungan1.getSaldo());
36. boolean hasilTransfer = tabungan1.transfer(tabungan2, 1500);
37. if (hasilTransfer) {
38. System.out.println("Transfer berhasil dari Tabungan 1 ke Tabungan 2");
39. System.out.println("Saldo Tabungan 1 sekarang: " + tabungan1.getSaldo());
40. System.out.println("Saldo Tabungan 2 sekarang: " + tabungan2.getSaldo());
41. } else {
42. System.out.println("Transfer gagal, saldo Tabungan 1 tidak cukup!");
43. }
44. }
45. }

Penjelasan : Kode yang diberikan adalah sebuah kelas Java bernama "Tabungan" yang memiliki atribut saldo, konstruktor untuk inisialisasi saldo awal, metode untuk mendapatkan saldo, metode untuk menyimpan uang ke dalam tabungan, metode untuk mengambil uang dari tabungan, dan metode untuk mentransfer uang antara dua objek Tabungan.

Dalam metode main, dua objek Tabungan dibuat dengan saldo awal masing-masing 5000 dan 3000. Selanjutnya, dilakukan penyimpanan uang sebesar 2000 ke dalam Tabungan pertama. Hasil transfer sejumlah 1500 dari Tabungan pertama ke Tabungan kedua dievaluasi, dan jika berhasil, saldo kedua Tabungan dicetak. Jika transfer gagal karena saldo tidak mencukupi, pesan kesalahan akan ditampilkan.

Kode ini menggambarkan penggunaan kelas Tabungan untuk mengelola saldo, menyimpan uang, mengambil uang, dan mentransfer uang antara dua objek Tabungan.