**TUGAS 3**

**MATA KULIAH ALGORITMA DAN DASAR PEMROGRAMAN SEMESTER GENAP TAHUN AKADEMIK 2023-2024**



**Oleh :**

**WILLY ACHMAD NURANI**

**(4123024)**

**JURUSAN SISTEM INFORMASI**

**UNIVERSITAS PESANTREN TINGGI DARUL 'ULUM JOMBANG**

**2024**

1. Class Rekening Bank

Attribute:

-nomorRekening: String

-saldo: double

Method:

+AkunBank(nomorRekening: String, saldo: double): void

+getNomorRekening(): String

+getSaldo(): double

+simpan(jumlah: double): void

+ambil(jumlah: double): void

+transfer(jumlah: double, targetAkun: AkunBank): void

package rekeningbank;

/\*\*

\*

\* @author LENOVO WILLY

\*/

public class RekeningBank {

private String nomorRekening;

private double saldo;

public RekeningBank(String nomorRekening, double saldo) {

this.nomorRekening = nomorRekening;

this.saldo = saldo;

}

public String getNomorRekening() {

return nomorRekening;

}

public double getSaldo() {

return saldo;

}

public void deposit(double jumlah) {

saldo += jumlah;

}

public boolean withdraw(double jumlah) {

if (saldo >= jumlah) {

saldo -= jumlah;

return true;

} else {

return false;

}

}

public boolean transfer(double jumlah, RekeningBank rekeningTujuan) {

if (withdraw(jumlah)) {

rekeningTujuan.deposit(jumlah);

return true;

} else {

return false;

}

}

public static void main(String[] args) {

RekeningBank rekening1 = new RekeningBank("123456", 5000.00);

RekeningBank rekening2 = new RekeningBank("789101", 200.00);

System.out.println("Saldo rekening 1: " + rekening1.getSaldo());

System.out.println("Saldo rekening 2: " + rekening2.getSaldo());

rekening1.withdraw(700.00);

rekening2.deposit(700.00);

System.out.println("Saldo rekening 1 setelah penarikan: " + rekening1.getSaldo());

System.out.println("Saldo rekening 2 setelah penyetoran: " + rekening2.getSaldo());

rekening1.transfer(400.00, rekening2);

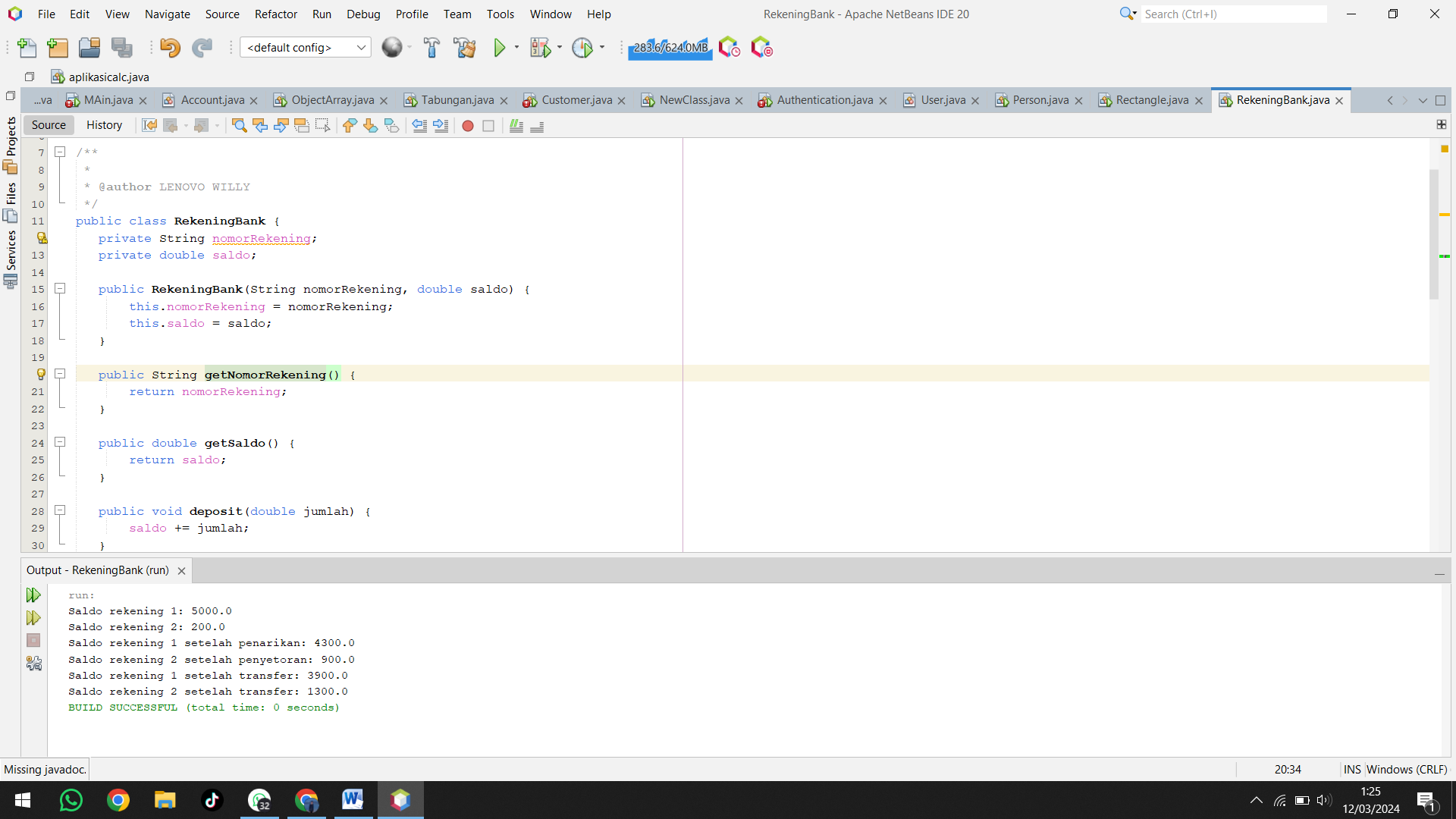
System.out.println("Saldo rekening 1 setelah transfer: " + rekening1.getSaldo());

System.out.println("Saldo rekening 2 setelah transfer: " + rekening2.getSaldo());

}

}

**Output :**



**Analisa :**

Kelas RekeningBank digunakan untuk merepresentasikan rekening bank dengan properti nomor rekening dan saldo. Berikut adalah analisisnya:

* Constructors: Terdapat satu konstruktor yang menerima dua parameter, yaitu nomor rekening dan saldo awal, dan menginisialisasi properti-properti tersebut.
* Getter Methods: Terdapat metode getNomorRekening() dan getSaldo() yang mengembalikan nilai nomor rekening dan saldo.
* deposit() Method: Metode deposit() digunakan untuk menambahkan jumlah tertentu ke saldo rekening.
* withdraw() Method: Metode withdraw() digunakan untuk menarik sejumlah uang dari saldo rekening, dengan melakukan pengecekan apakah saldo mencukupi sebelum penarikan dilakukan.
* transfer() Method: Metode transfer() digunakan untuk mentransfer sejumlah uang dari rekening saat ini ke rekening lain. Metode ini memanfaatkan metode withdraw() untuk melakukan penarikan dari rekening saat ini dan metode deposit() pada rekening tujuan untuk menyetor uang yang ditarik.
* Main Method: Digunakan untuk menguji fungsionalitas kelas. Dua objek RekeningBank dibuat dengan saldo awal yang berbeda, kemudian berbagai operasi seperti penarikan, penyetoran, dan transfer uang antara kedua rekening dilakukan. Informasi saldo setiap rekening dicetak sebelum dan setelah setiap operasi.

Kelas ini memberikan cara sederhana untuk mengelola rekening bank, termasuk melakukan operasi deposit, penarikan, dan transfer.

1. Class: Person

Attribute:

-nama: String

-umur: int

-alamat: String

Method:

+Person(nama: String, umur: int, alamat: String): void

+getNama(): String

+getUmur(): int

+getAlamat(): String

+setNama(nama: String): void

+setUmur(umur: int): void

+setAlamat(alamat: String): void

package person;

/\*\*

\*

\* @author LENOVO WILLY

\*/

public class Person {

private String nama;

private int umur;

private String alamat;

public Person(String nama, int umur, String alamat) {

this.nama = nama;

this.umur = umur;

this.alamat = alamat;

}

public String getName() {

return nama;

}

public void setName(String nama) {

if (nama == null || nama.trim().isEmpty()) {

throw new IllegalArgumentException("Name cannot be null or empty");

}

this.nama = nama;

}

public int getAge() {

return umur;

}

public void setAge(int umur) {

if (umur < 0) {

throw new IllegalArgumentException("Age cannot be negative");

}

this.umur = umur;

}

public String getAddress() {

return alamat;

}

public void setAddress(String alamat) {

if (alamat == null || alamat.trim().isEmpty()) {

throw new IllegalArgumentException("Address cannot be null or empty");

}

this.alamat = alamat;

}

@Override

public String toString() {

return "Person{" +

"nama='" + nama + '\'' +

", umur=" + umur +

", alamat='" + alamat + '\'' +

'}';

}

public static void main(String[] args) {

Person person = new Person("Nayip", 28, "dahna blok a 10");

System.out.println(person);

person.setName("Mian");

person.setAge(24);

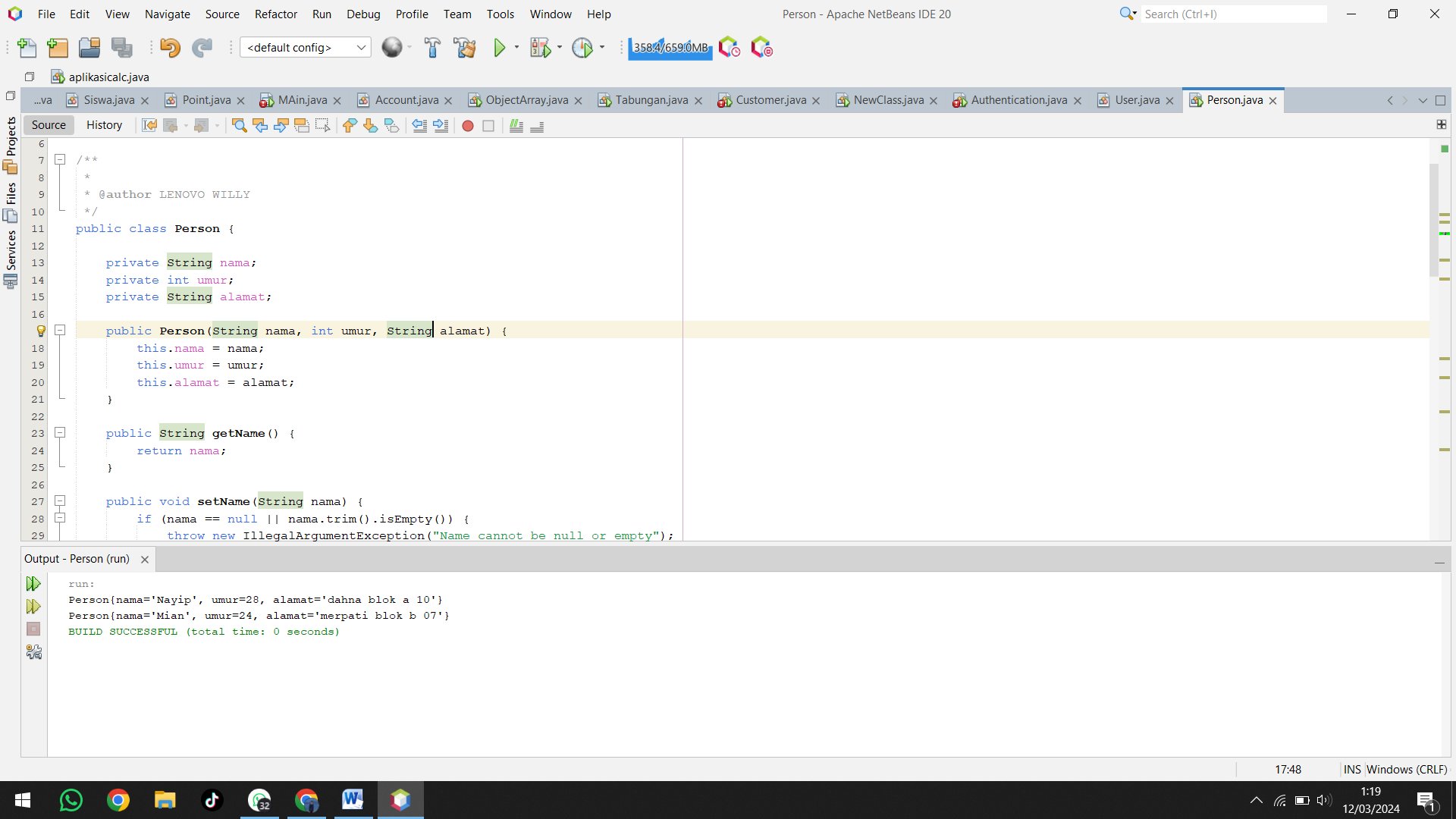
person.setAddress("merpati blok b 07");

System.out.println(person);

}

}

**Output :**



**Analisa :**

Kelas Person adalah representasi dari individu dengan properti seperti nama, umur, dan alamat. Ini memiliki metode untuk mengatur dan mendapatkan nilai-nilai ini dengan validasi yang sesuai.

* Constructors: Kelas ini memiliki satu konstruktor yang menerima nilai untuk nama, umur, dan alamat, dan menginisialisasi properti-properti tersebut.
* Getter dan Setter Methods: Ada metode getter dan setter untuk setiap properti (getName(), setName(), getAge(), setAge(), getAddress(), setAddress()). Metode setter memeriksa validitas input sebelum mengubah nilai properti. Jika input tidak valid (misalnya nama atau alamat kosong, atau umur negatif), pengecualian dilemparkan.
* toString() Method: Ini dioverride untuk memberikan representasi String dari objek Person, yang mencakup nama, umur, dan alamat.
* Main Method: Ini adalah metode main() yang digunakan untuk menguji fungsionalitas kelas. Sebuah objek Person dibuat dengan nilai-nilai awal, dicetak, kemudian nilai-nilainya diubah menggunakan setter, dan objek tersebut dicetak lagi untuk memeriksa apakah perubahan berhasil.
* Exception Handling: Pengecualian dilemparkan jika input yang diberikan ke metode setter tidak valid.

Secara keseluruhan, kelas ini memberikan cara yang baik untuk merepresentasikan individu dan memastikan bahwa data yang dimasukkan valid.

1. Class Persegi Panjang

Attribute:

-lebar: int

-tinggi: int

Method:

+PersegiPanjang(lebar: int, tinggi: int): void

+lebar(): int

+tinggi(): int

+setLebar(lebar: int): void

+setTinggi(tinggi: int): void

+getLuas(): int

+getKeliling(): int

package rectangle;

/\*\*

\*

\* @author LENOVO WILLLY

\*/

public class Rectangle {

private int lebar;

private int tinggi;

public Rectangle(int lebar, int tinggi) {

this.lebar = lebar;

this.tinggi = tinggi;

}

public int getWidth() {

return lebar;

}

public int getHeight() {

return tinggi;

}

public int getArea() {

return lebar \* tinggi;

}

public int getPerimeter() {

return 2 \* (lebar + tinggi);

}

public void setWidth(int lebar) {

this.lebar = lebar;

}

public void setHeight(int tinggi) {

this.tinggi = tinggi;

}

public static void main(String[] args) {

Rectangle rectangle = new Rectangle(15, 5);

System.out.println("lebar: " + rectangle.getWidth());

System.out.println("tinggi: " + rectangle.getHeight());

System.out.println("luas: " + rectangle.getArea());

System.out.println("keliling: " + rectangle.getPerimeter());

rectangle.setWidth(10);

rectangle.setHeight(25);

System.out.println("Updated lebar: " + rectangle.getWidth());

System.out.println("Updated tinggi: " + rectangle.getHeight());

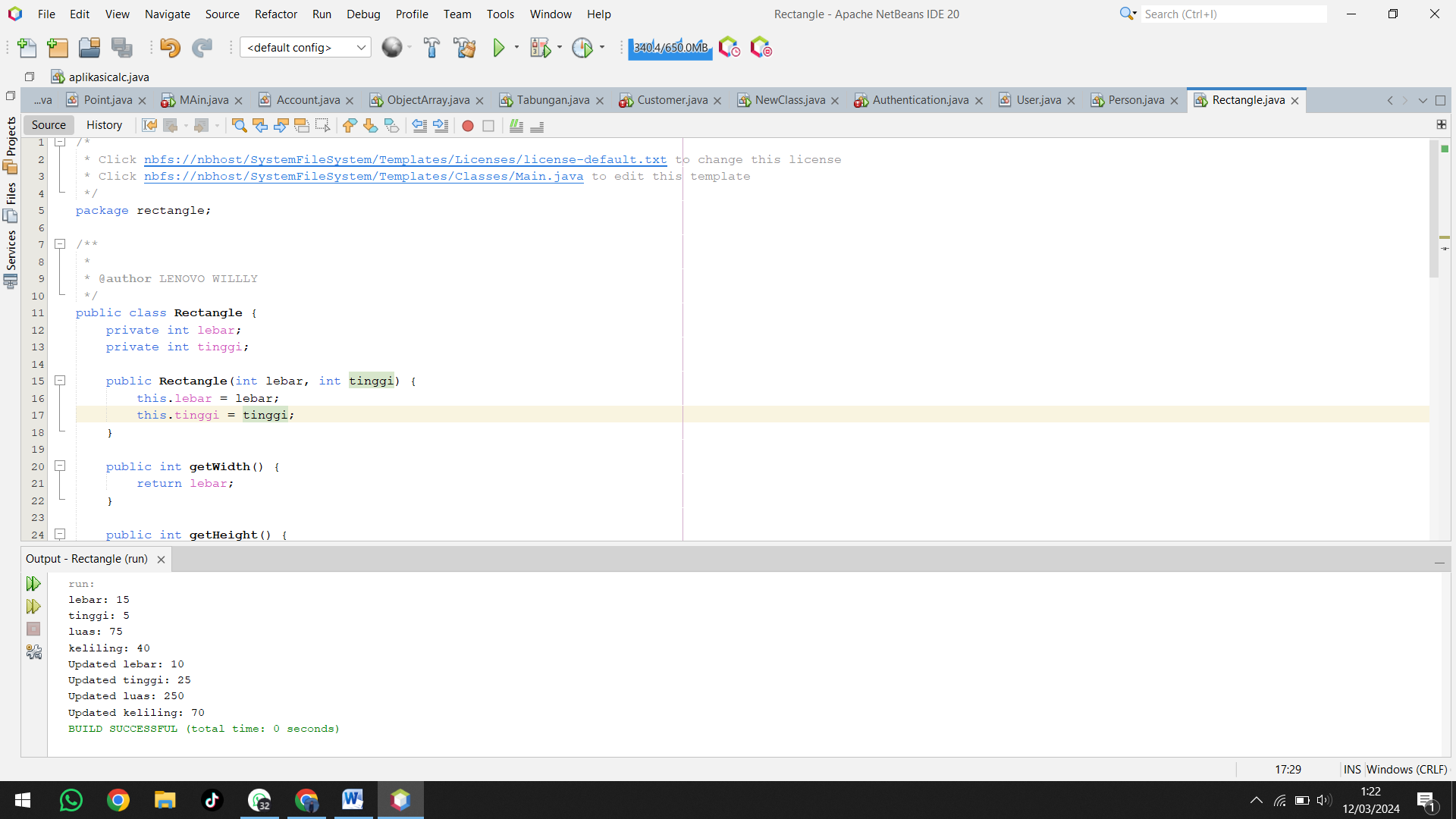
System.out.println("Updated luas: " + rectangle.getArea());

System.out.println("Updated keliling: " + rectangle.getPerimeter());

}

}

**Output :**



**Analisa :**

Kelas Rectangle adalah representasi sederhana dari persegi panjang dengan properti lebar dan tinggi. Berikut adalah analisisnya:

* Constructors: Terdapat satu konstruktor yang menerima dua parameter, yaitu lebar dan tinggi, dan menginisialisasi properti-properti tersebut.
* Getter Methods: Terdapat metode getWidth() dan getHeight() yang mengembalikan nilai lebar dan tinggi persegi panjang, masing-masing.
* Area and Perimeter Calculation Methods: Terdapat metode getArea() dan getPerimeter() yang menghitung dan mengembalikan luas dan keliling persegi panjang, berdasarkan lebar dan tinggi.
* Setter Methods: Terdapat metode setWidth() dan setHeight() yang mengatur nilai lebar dan tinggi persegi panjang.
* Main Method: Digunakan untuk menguji fungsionalitas kelas. Sebuah objek Rectangle dibuat dengan nilai lebar dan tinggi awal, kemudian informasi tentang persegi panjang tersebut dicetak. Nilai lebar dan tinggi kemudian diubah menggunakan metode setter, dan informasi tentang persegi panjang yang telah diperbarui dicetak kembali.

Kelas ini memberikan cara sederhana untuk mengelola persegi panjang, termasuk pengukuran luas dan kelilingnya.