**TUGAS 8**

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

****

**Oleh:**

**AINUR ROHMAN**

**HARDINI**

**4123016**

**PROGRAM STRATA-1 JURUSAN SISTEM INFORMASI**

**UNIVERSITAS PESANTREN TINGGI DARUL 'ULUM JOMBANG**

**Latihan 1. Overriding**

source code:

/\*\*

\*

\* @author AINUR

\*/

public class Base{

private void amethod(int iBase){

System.out.println("Base.amethod");

}

}

class Over extends Base{

public static void main(String argv[]){

Over o = new Over();

int iBase=0;

o.amethod(iBase);

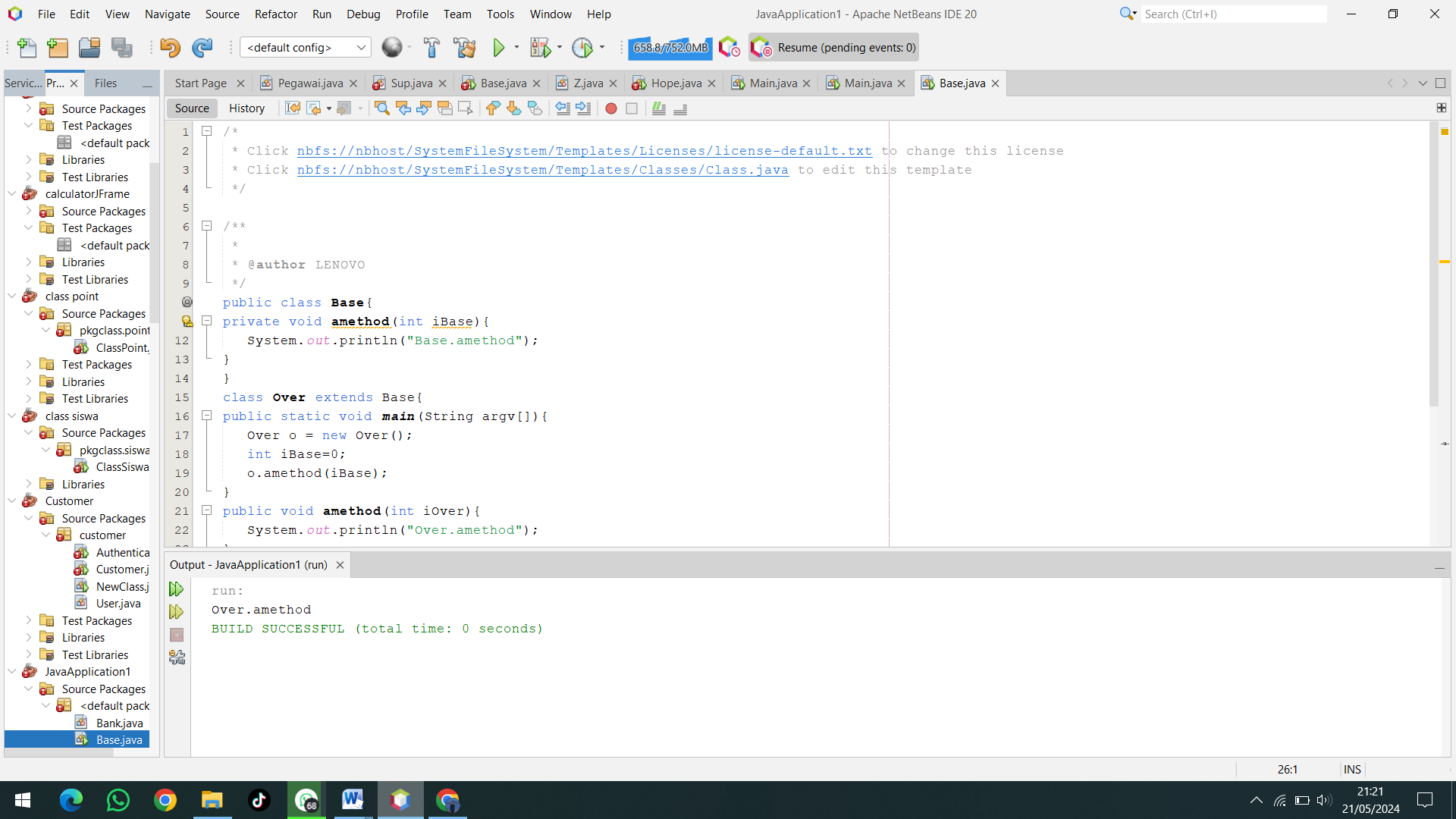
}

public void amethod(int iOver){

System.out.println("Over.amethod");

}

}



**Latihan 2. Overloading**

Apa yang terjadi bila program berikut ini dikompile dan dijalankan? Jelaskan !

Source code:

/\*\*

\*

\* @author AINUR

\*/

class MyParent {

int x, y;

MyParent(int x, int y){

this.x = x;

this.y = y;

}

public int addMe(int x, int y){

return this.x + x + y + this.y;

}

public int addMe(MyParent myPar){

return addMe(myPar.x, myPar.y);

}

}

class MyChild extends MyParent{

int z;

MyChild (int x, int y, int z) {

super(x,y);

this.z = z;

}

public int addMe(int x, int y, int z){

return this.x + x + this.y + y + this.z + z;

}

public int addMe(MyChild myChi){

return addMe(myChi.x, myChi.y, myChi.z);

}

public int addMe(int x, int y){

return this.x + x + this.y + y;

}

}

public class MySomeOne{

public static void main(String args[]){

MyChild myChi = new MyChild(10, 20, 30);

MyParent myPar = new MyParent(10, 20);

int x = myChi.addMe(10, 20, 30);

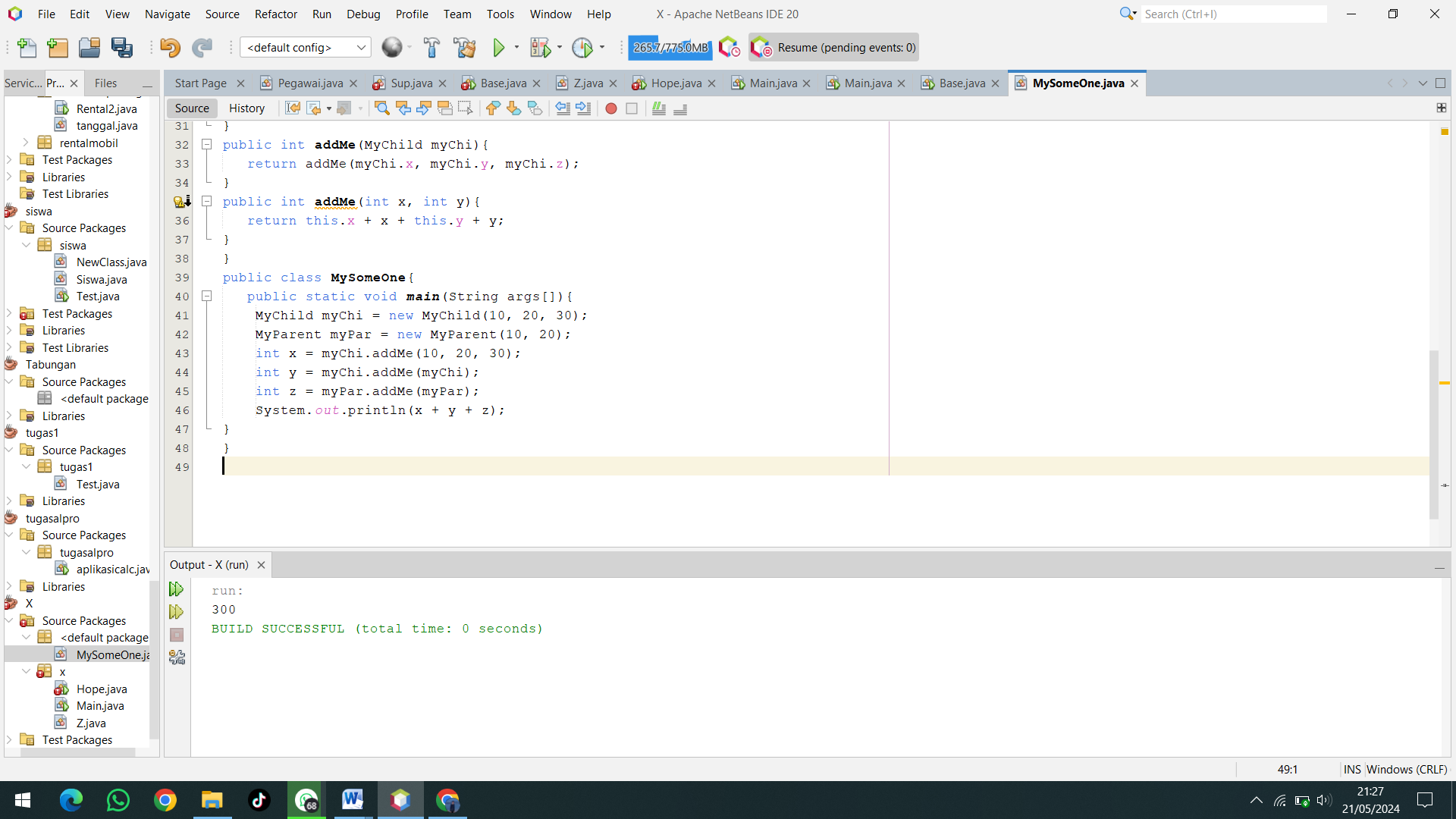
int y = myChi.addMe(myChi);

int z = myPar.addMe(myPar);

System.out.println(x + y + z);

}

}



**Latihan 3. Overloading**

Source code:

/\*\*

\*

\* @author AINUR

\*/

public class MyClass{

void myMethod(int i) {

System.out.println("int version");

}

void myMethod(String s) {

System.out.println("String version");

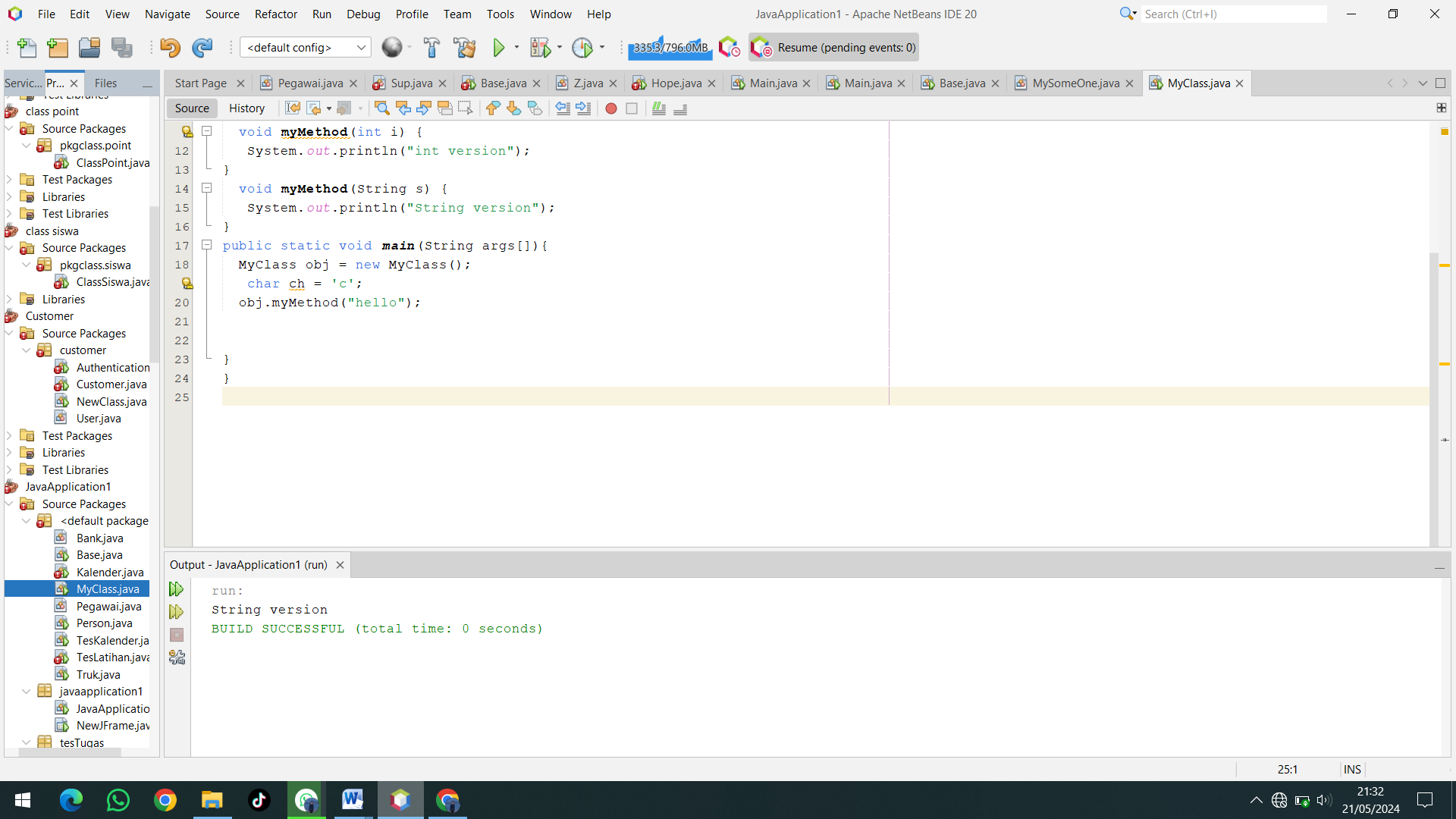
}

public static void main(String args[]){

MyClass obj = new MyClass();

char ch = 'c';

obj.myMethod("ch");



**Latihan 4. Mengimplementasikan UML class diagram dalam program**

source code:

/\*\*

\*

\* @author AINUR

\*/

class Orang {

private String nama;

private int umur;

public Orang(String nama) {

this.nama = nama;

}

public Orang(String nama, int umur) {

this.nama = nama;

this.umur = umur;

}

public void Info() {

System.out.println("Nama: " + nama);

if (umur != 0) {

System.out.println("Umur: " + umur);

}

}

}

class Dosen extends Orang {

private int nip;

public Dosen(String nama) {

super(nama);

}

public Dosen(String nama, int nip) {

super(nama);

this.nip = nip;

}

public Dosen(String nama, int nip, int umur) {

super(nama, umur);

this.nip = nip;

}

@Override

public void Info() {

super.Info();

System.out.println("NIP: " + nip);

}

}

public class TesLatihan {

public static void main(String args[]) {

System.out.println("Memasukkan identitas dosen 1 : Agus");

Dosen dosen1 = new Dosen("Agus");

System.out.println("Memasukkan identitas dosen 2 : Budi, NIP. 1458");

Dosen dosen2 = new Dosen("Budi", 1458);

System.out.println("Memasukkan identitas dosen 3 : Iwan, NIP. 1215, umur 47");

Dosen dosen3 = new Dosen("Iwan", 1215, 47);

System.out.println();

dosen1.Info();

System.out.println();

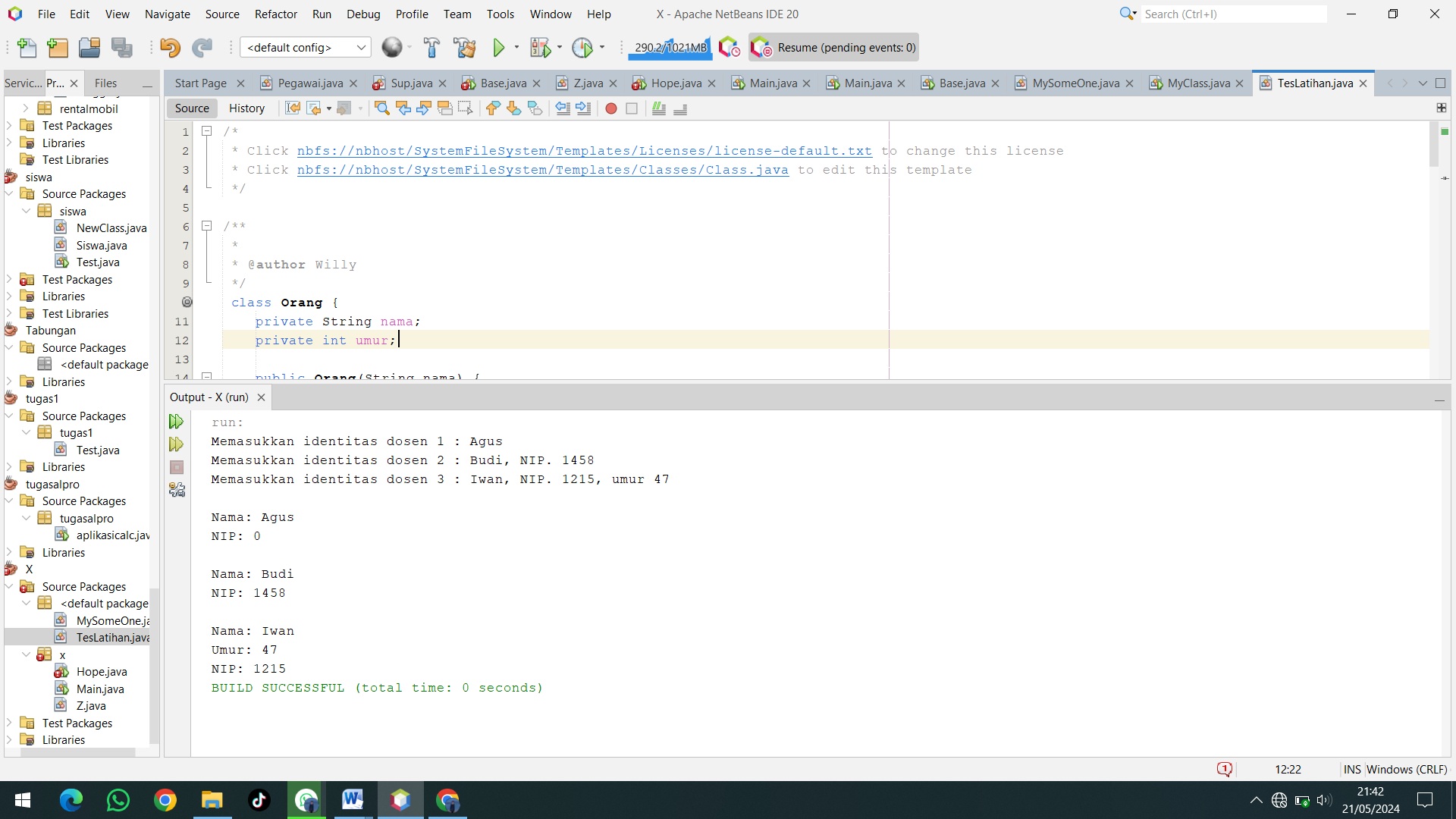
dosen2.Info();

System.out.println();

dosen3.Info();

}

}



* Inheritance memungkinkan kelas Dosen untuk mewarisi atribut dan metode dari kelas Orang.
* Overriding digunakan untuk memberikan implementasi spesifik dari metode Info() dalam kelas Dosen.
* Constructors di kelas turunan memanggil konstruktor kelas induk menggunakan super(), tetapi tidak diwariskan secara langsung.

**Tugas 1**

source code:

/\*\*

\*

\* @author AINUR

\*/

class RerataNilai {

public double average(double... angka) {

double hitung = 0;

for (double Angka : angka) {

hitung += Angka;

}

return hitung / angka.length;

}

}

public class TesTugas1{

public static void main(String args[]){

RerataNilai rn = new RerataNilai();

System.out.println("Rerata nilai 21 dan 13 adalah : " + rn.average(21, 13));

System.out.println("Rerata nilai 19.3 dan 39.5 adalah : " + rn.average(19.3, 39.5));

System.out.println("Rerata nilai 123, 567 dan 744 adalah : " + rn.average(123, 567, 744));

}

}

* Kelas RerataNilai:

Kelas ini memiliki satu metode average yang menerima sejumlah argumen variabel (double... angka).

Metode average menghitung rerata dari nilai-nilai yang diberikan.

* Metode average:

public double average(double... angka): Metode ini menggunakan fitur varargs di Java, yang memungkinkan metode untuk menerima sejumlah argumen yang tidak ditentukan sebelumnya.

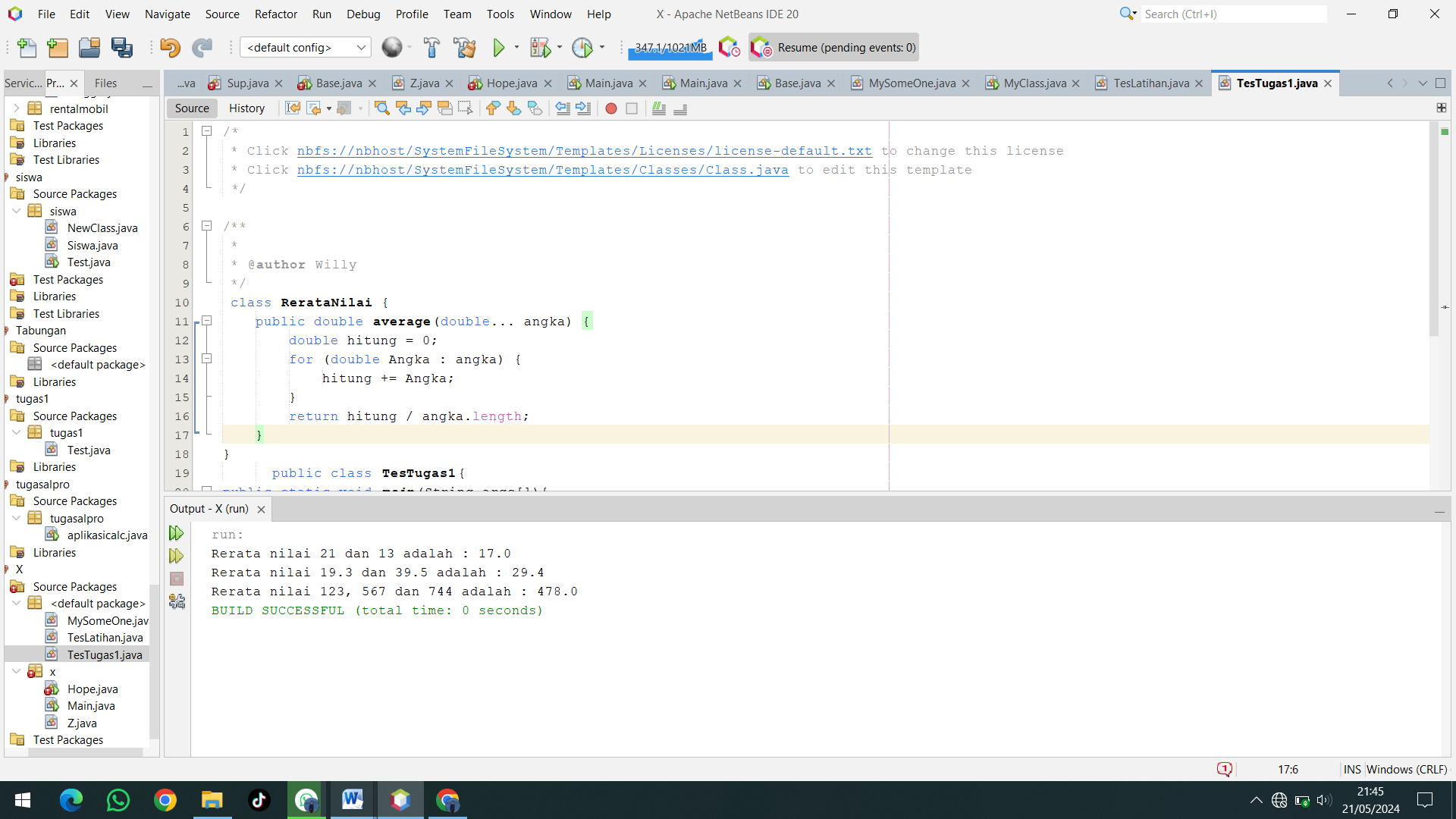
double hitung = 0;: Inisialisasi variabel hitung untuk menyimpan jumlah total nilai.

for (double Angka : angka) { hitung += Angka; }: Loop untuk menjumlahkan semua nilai yang diberikan sebagai argumen.

return hitung / angka.length;: Mengembalikan hasil pembagian total jumlah nilai dengan jumlah elemen yang diberikan (rata-rata).

* Kelas TesTugas1:

Kelas ini berisi metode main yang digunakan untuk menguji metode average dari kelas RerataNilai.



**Tugas 2**

source code:

\*\*

\*

\* @author AINUR

\*/

class Katak {

int umur;

String nama;

public Katak(int umur, String nama) {

this.umur = umur;

this.nama = nama;

}

public String caraBergerak() {

// Implementasi method ini dapat disesuaikan dengan cara bergerak kura-kura

return " Melompat ";

}

}

class Kecebong extends Katak {

double panjangEkor;

public Kecebong(int umur, String nama, double panjangEkor) {

super(umur, nama);

this.panjangEkor = panjangEkor;

}

// Override method caraBergerak dari class Katak

@Override

public String caraBergerak() {

// Implementasi method ini dapat disesuaikan dengan cara bergerak kecebong

return " Berenang ";

}

}

public class Main {

public static void main(String[] args) {

Katak katak = new Katak(5, "Froggy");

System.out.println("Nama: " + katak.nama + ", Umur: " + katak.umur + ", Cara Bergerak: " + katak.caraBergerak());

Kecebong kecebong = new Kecebong(2, "Junior Frog", 10);

System.out.println("Nama: " + kecebong.nama + ", Umur: " + kecebong.umur + ", Panjang Ekor: " + kecebong.panjangEkor + ", Cara Bergerak: " + kecebong.caraBergerak());

}

}

* Inheritance memungkinkan Kecebong untuk mewarisi atribut dan metode dari Katak.
* Overriding digunakan di Kecebong untuk memberikan implementasi spesifik dari metode caraBergerak().
* Program mencetak informasi yang tepat tentang objek katak dan kecebong, menunjukkan bagaimana mereka menggunakan metode yang di-overridden dan warisan dari superclass.

