**TUGAS 7**

**MATA KULIAH ALGORITMA DAN DASAR PEMROGRAMAN SEMESTERGENAPTAHUN AKADEMIK2023-2024**



**Oleh**

**Muhammad Syarif H**

**(4123025)**

**JURUSAN SISTEM INFORMASI**

**UNIVERSITAS PESANTREN TINGGI DARUL 'ULUMJOMBANG**

**2024**

C. TUGAS PENDAHULUAN

1. Ada berapa modifier untukpengontrolanakses? Jelaskanmasing-masing!
2. Apakahkegunaan kata kunci super? Jelaskan !
3. Apakah yang dimaksuddengankonstruktortidakdiwariskan?

Jawaban:

1. Ada empat modifier untukpengontrolanaksesdalam Java:

* public: Member yang dideklarasikansebagai public dapatdiaksesdarimanasaja, baikdaridalamkelasitusendiri, kelas yang berbedadalampaket yang sama, ataudarikelas lain di luarpaket.
* private: Member yang dideklarasikansebagai private hanyadapatdiakses di dalamkelasitusendiri. Tidakdapatdiaksesdarikelaslain, bahkankelasdalampaket yang samasekalipun.
* protected: Member yang dideklarasikansebagai protected dapatdiaksesdarikelasitusendiri, kelas yang berbedadalampaket yang sama, dandarikelasturunan (subclass) bahkanjika subclass berada di luarpaket.
* default (tidakmenggunakan modifier): Jikatidakada modifier yang ditentukan, maka member tersebutakanmemilikiakses default, yang artinya member tersebutdapatdiakseshanyaolehkelas yang beradadalampaket yang sama.

1. Kata kunci super digunakanuntukmerujukkekelasinduk (superclass) darisuatukelas. Kegunaanutamadari kata kunci super adalah:

MemanggilKonstruktorKelasInduk: super() digunakandalamkonstruktorkelasturunanuntukmemanggilkonstruktordarikelasinduk. Hal inimemungkinkaninisialisasiatribut yang diwariskandarikelasinduk.

Mengakses Member KelasInduk: super digunakanuntukmengakses member (atributataumetode) darikelasinduksaatadanama yang samaantarakelasindukdankelasturunan. Denganmenggunakan super, kitabisamembedakanantaraatributataumetodedarikelasturunandankelasinduk.

1. Konstruktortidakdiwariskanberartibahwakonstruktordarikelasinduktidaksecaraotomatisditurunkankepadakelasturunannya. Meskipunkelasturunanakanmenggunakankonstruktordarikelasindukuntukmenginisialisasiatribut yang diwarisi, namunkonstruktoritusendiritidakditurunkansecaralangsung.Sebagaicontoh:

class Base{

Base(){

int i = 100;

System.out.println(i);

}

}

public class Pri extends Base{

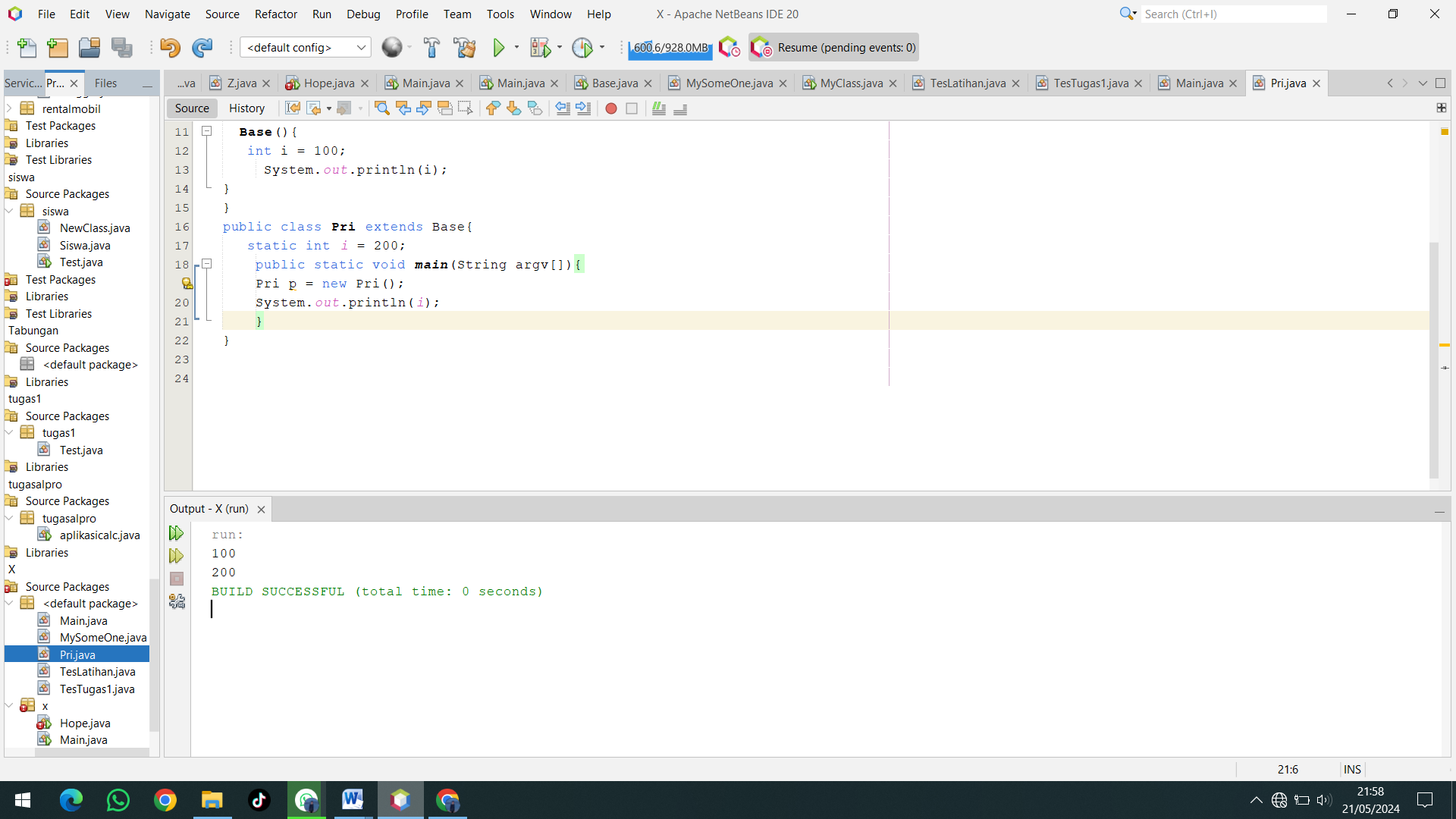
staticint i = 200;

public static void main(String argv[]){

Pri p = new Pri();

System.out.println(i);

}



**Latihan 1**

package pkg7;

/\*\*

\*

\* @author Willy

\*/

class Base{

Base(int i){

System.out.println("base constructor");

}

Base(){

this(42);

}

}

public class Sup extends Base{

public Sup(){

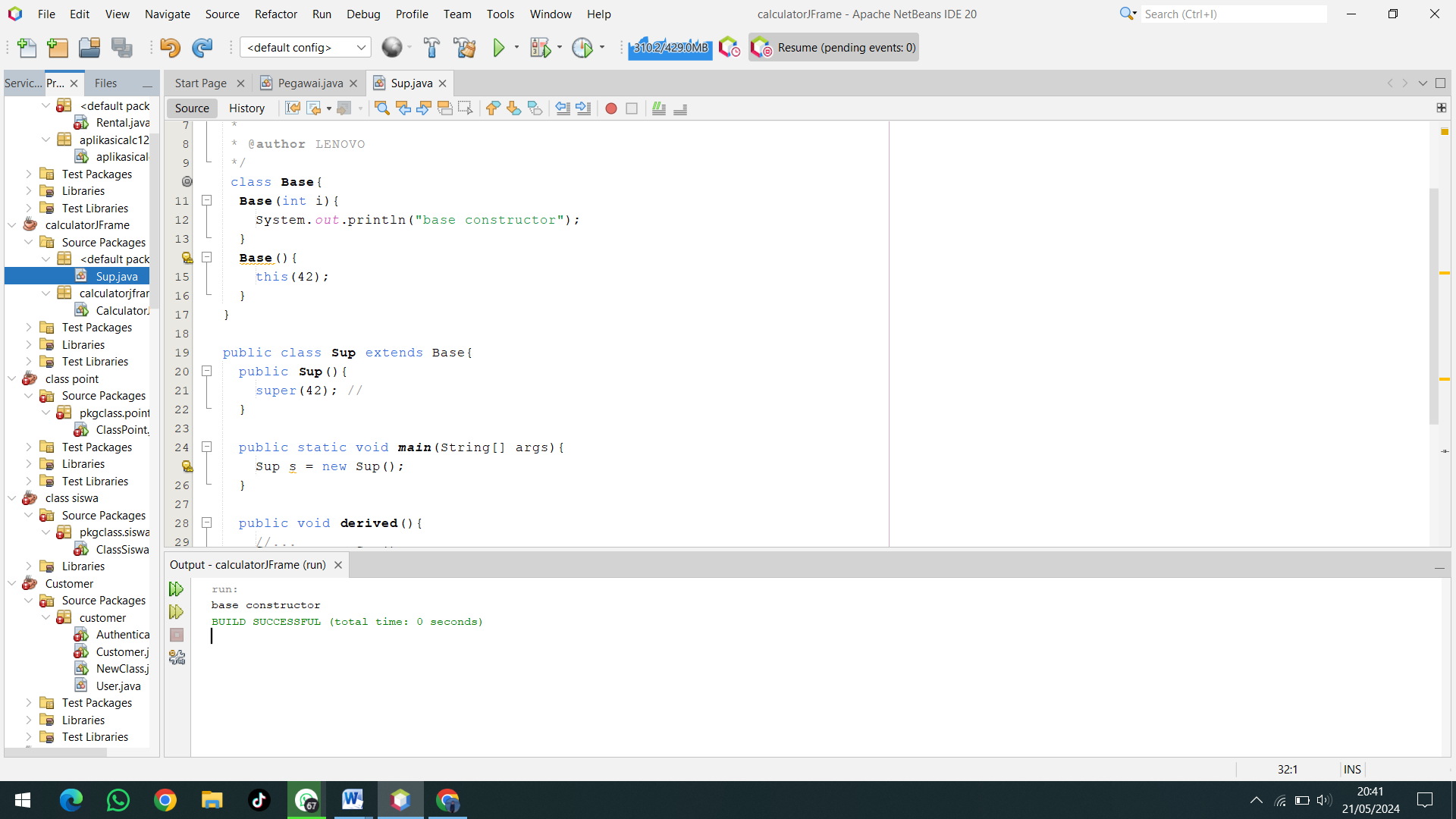
super(42); //

}

public static void main(String[] args){

Sup s = new Sup();

}



**Latihan 2**

Private class Base{

Base(){

int i = 100;

System.out.println(i);

}

}

public class Pri extends Base{

static int i = 200;

public static void main(String argv[]){

Pri p = new Pri();

System.out.println(i);

}

Kelas Base

Kelas Base adalah sebuah kelas privat yang memiliki sebuah konstruktor tanpa parameter. Konstruktor ini melakukan dua hal:

Menginisialisasi sebuah variabel lokal i dengan nilai 100.

Mencetak nilai i ke layar menggunakan System.out.println(i).

Kelas Pri

Kelas Pri adalah sebuah kelas publik yang extends kelas Base. Kelas ini memiliki dua komponen:

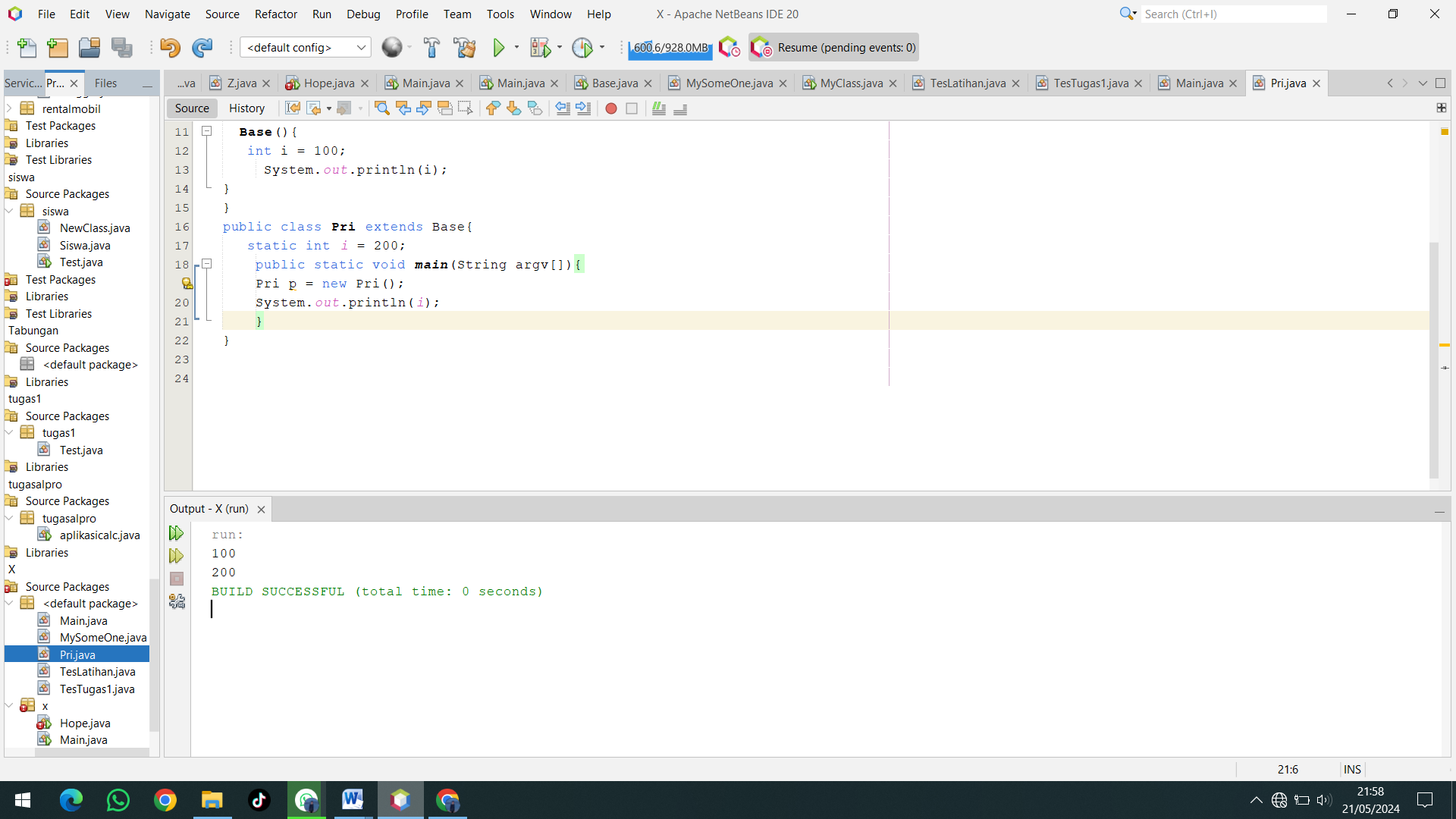
Sebuah variabel statis i dengan nilai 200.

Sebuah metode main yang merupakan entry point dari program.

Dalam metode main, terdapat dua baris kode:

Membuat sebuah instance dari kelas Pri dan menyimpannya dalam variabel p.

Mencetak nilai variabel statis i ke layar menggunakan System.out.println(



**Latihan 3**

package x;

/\*\*

\*

\* @author Willy

\*/

class X{

Y b = new Y();

X(){

System.out.print("X");

}

}

public class Z extends X{

Y y = new Y();

Z(){

System.out.print("Z");

}

public static void main(String[] args){

new Z();

}

}

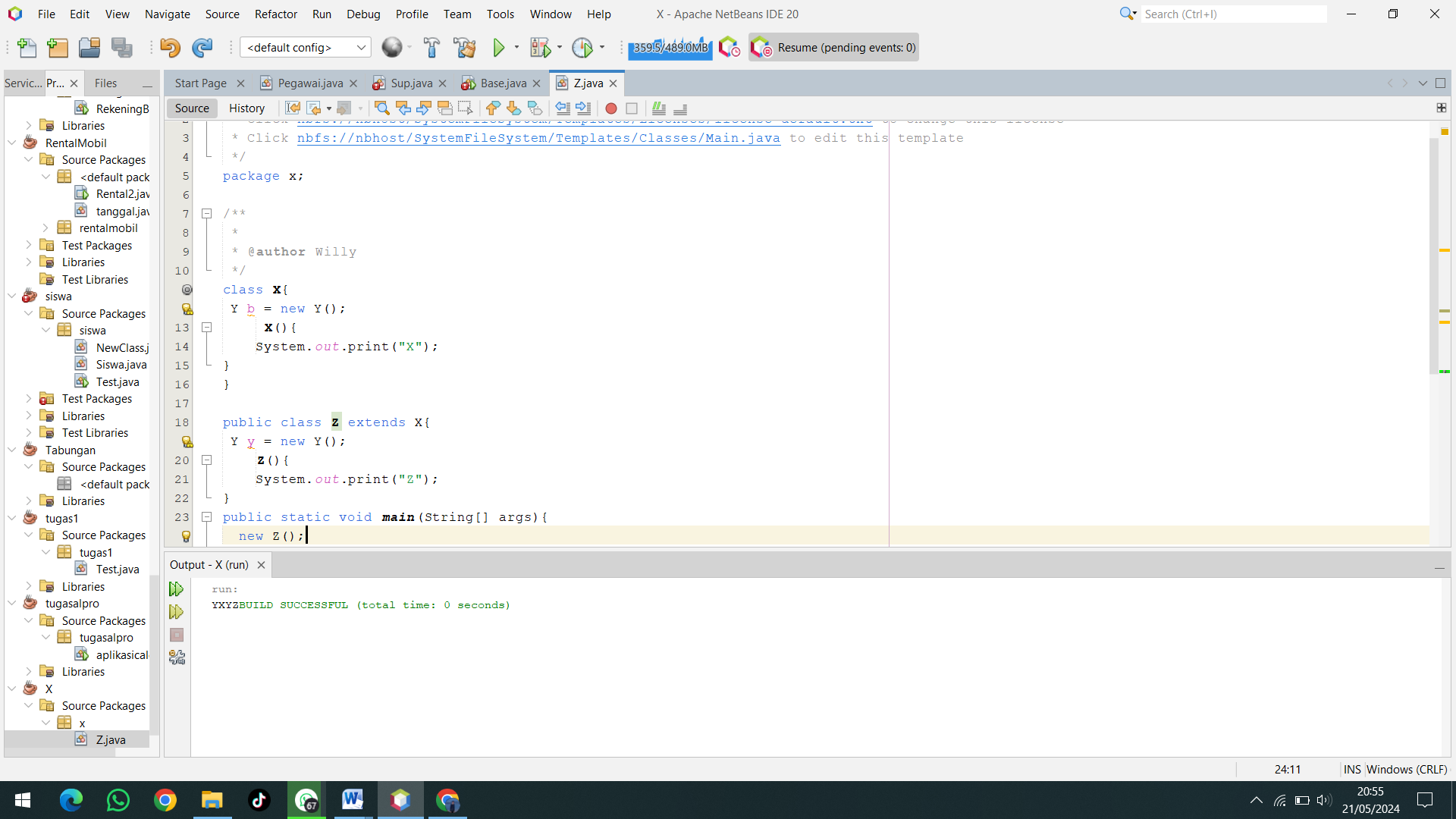
class Y{

Y(){

System.out.print("Y");

}

}



**Latihan 4**

/\*\*

\*

\* @author Willy

\*/

public class Hope{

public static void main(String argv[]){

Hope h = new Hope();

}

protected Hope(){

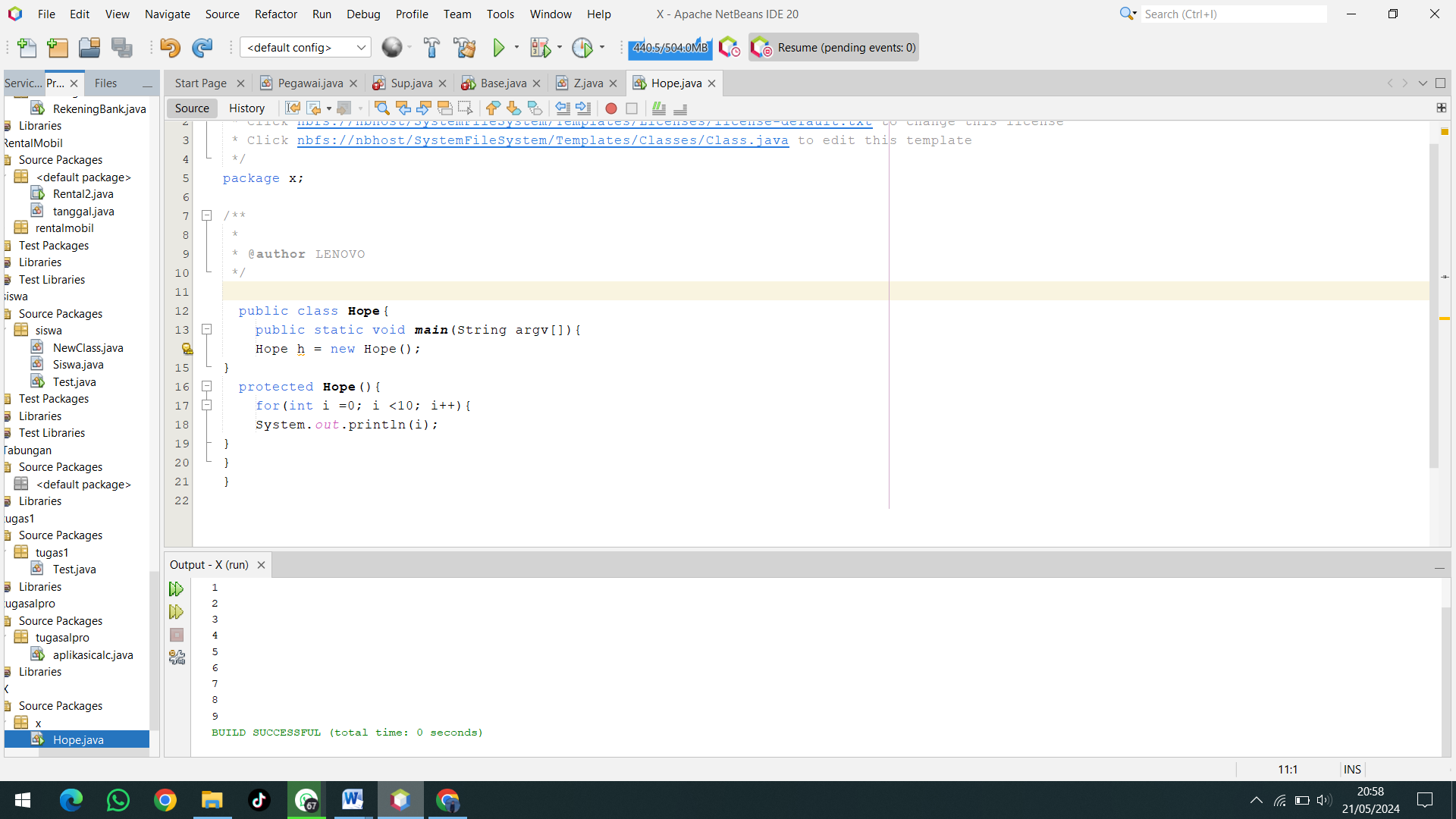
for(int i =0; i <10; i++){

System.out.println(i);

}

}

}



**Tugas**

Souce code:

package main;

/\*\*

\*

\* @author Willy

\*/

abstract class Vehicle {

private final int numberOfWheels;

private final String color;

public Vehicle(int numberOfWheels, String color) {

this.numberOfWheels = numberOfWheels;

this.color = color;

}

public int getNumberOfWheels() {

return numberOfWheels;

}

public String getColor() {

return color;

}

@Override

public abstract String toString();

}

class Bicycle1 extends Vehicle {

private final int numberOfSaddles;

private final int numberOfGears;

public Bicycle1(int numberOfWheels, String color, int numberOfSaddles, int numberOfGears) {

super(numberOfWheels, color);

this.numberOfSaddles = numberOfSaddles;

this.numberOfGears = numberOfGears;

}

public int getNumberOfSaddles() {

return numberOfSaddles;

}

public int getNumberOfGears() {

return numberOfGears;

}

@Override

public String toString() {

return "Sepeda : jumlah roda " + getNumberOfWheels() + ", warna " + getColor() + ", jumlah sadle " + numberOfSaddles + ", Jumlah gear " + numberOfGears + " biji ";

}

}

class Bicycle extends Vehicle {

private final int numberOfSaddles;

private final int numberOfGears;

public Bicycle(int numberOfWheels, String color, int numberOfSaddles, int numberOfGears) {

super(numberOfWheels, color);

this.numberOfSaddles = numberOfSaddles;

this.numberOfGears = numberOfGears;

}

public int getNumberOfSaddles() {

return numberOfSaddles;

}

public int getNumberOfGears() {

return numberOfGears;

}

@Override

public String toString() {

return "Sepeda : jumlah roda " + getNumberOfWheels() + ", warna " + getColor() + ", jumlah sadle " + numberOfSaddles + ", Jumlah gear " + numberOfGears + " biji ";

}

}

class Truck extends Vehicle {

private final int maxLoad;

private final String fuelType;

private final int engineCapacity;

public Truck(int numberOfWheels, String color, String fuelType , int engineCapacity,int maxLoad) {

super(numberOfWheels, color);

this.maxLoad = maxLoad;

this.fuelType = fuelType;

this.engineCapacity = engineCapacity;

}

public String getFuelType() {

return fuelType;

}

public int getMaxLoad() {

return maxLoad;

}

public int getEngineCapacity() {

return engineCapacity;

}

@Override

public String toString() {

return "Truck 1 : jumlah roda " + getNumberOfWheels() + ", warna " + getColor() + ", jenis bahan bakar " + fuelType + ", kapasitas mesin " + engineCapacity + "CC, kapasitas maksimal " + maxLoad + " Kg ";

}

}

class Truck1 extends Vehicle {

private final int maxLoad;

private final String fuelType;

private final int engineCapacity;

public Truck1(int numberOfWheels, String color, String fuelType , int engineCapacity,int maxLoad) {

super(numberOfWheels, color);

this.maxLoad = maxLoad;

this.fuelType = fuelType;

this.engineCapacity = engineCapacity;

}

public String getFuelType() {

return fuelType;

}

public int getMaxLoad() {

return maxLoad;

}

public int getEngineCapacity() {

return engineCapacity;

}

@Override

public String toString() {

return "Truck 2 : jumlah roda " + getNumberOfWheels() + ", warna " + getColor() + ", jenis bahan bakar " + fuelType + ", kapasitas mesin " + engineCapacity + "CC, kapasitas maksimal " + maxLoad + " Kg ";

}

}

class Taxi extends Vehicle {

private final int baseFare;

private final int farePerKm;

private final String fuelType;

private final int engineCapacity;

public Taxi(int numberOfWheels, String color, String fuelType ,int engineCapacity, int baseFare, int farePerKm) {

super(numberOfWheels, color);

this.baseFare = baseFare;

this.farePerKm = farePerKm;

this.fuelType = fuelType;

this.engineCapacity = engineCapacity;

}

public String getFuelType() {

return fuelType;

}

public int getBaseFare() {

return baseFare;

}

public int getFarePerKm() {

return farePerKm;

}

public int getEngineCapacity() {

return engineCapacity;

}

@Override

public String toString() {

return "Taxi 1 : jumlah roda " + getNumberOfWheels() + ", warna " + getColor() + ", jenis bahan bakar " + fuelType + ", kapasitas mesin " + engineCapacity + "CC, tarif awal " + baseFare + ", tarif per Km " + farePerKm;

}

}

class Taxi1 extends Vehicle {

private final int baseFare;

private final int farePerKm;

private final String fuelType;

private final int engineCapacity;

public Taxi1(int numberOfWheels, String color, String fuelType ,int engineCapacity, int baseFare, int farePerKm) {

super(numberOfWheels, color);

this.baseFare = baseFare;

this.farePerKm = farePerKm;

this.fuelType = fuelType;

this.engineCapacity = engineCapacity;

}

public String getFuelType() {

return fuelType;

}

public int getBaseFare() {

return baseFare;

}

public int getFarePerKm() {

return farePerKm;

}

public int getEngineCapacity(){

return engineCapacity;

}

@Override

public String toString() {

return "Taxi 2 : jumlah roda " + getNumberOfWheels() + ", warna " + getColor() + ", jenis bahan bakar " + fuelType + ", kapasitas mesin " + engineCapacity + "CC, tarif awal " + baseFare + ", tarif per Km " + farePerKm;

}

}

public class Main {

public static void main(String[] args) {

Vehicle truck = new Truck(6, "Merah","solar",2000 ,1500);

Vehicle truck1 = new Truck1(4, "Kuning","solar",1500 ,1000);

Vehicle taxi = new Taxi(4, "orange","bensin",1500, 10000, 5000);

Vehicle taxi1 = new Taxi1(4, "biru","bensin",1000, 7000, 3500);

Vehicle bicycle = new Bicycle(3, "Hitam", 1, 2);

Vehicle bicycle1 = new Bicycle1(2, "Putih", 2, 5);

System.out.println(truck);

System.out.println(truck1);

System.out.println(taxi);

System.out.println(taxi1);

System.out.println(bicycle);

System.out.println(bicycle1);

}

}

**Output :**

