C. TUGAS PENDAHULUAN

1. Ada berapa modifier untuk pengontrolan akses? Jelaskan masing-masing!
2. Apakah kegunaan kata kunci super? Jelaskan !
3. Apakah yang dimaksud dengan konstruktor tidak diwariskan?

Jawaban:

1. Ada empat modifier untuk pengontrolan akses dalam Java:

* public: Member yang dideklarasikan sebagai public dapat diakses dari mana saja, baik dari dalam kelas itu sendiri, kelas yang berbeda dalam paket yang sama, atau dari kelas lain di luar paket.
* private: Member yang dideklarasikan sebagai private hanya dapat diakses di dalam kelas itu sendiri. Tidak dapat diakses dari kelas lain, bahkan kelas dalam paket yang sama sekalipun.
* protected: Member yang dideklarasikan sebagai protected dapat diakses dari kelas itu sendiri, kelas yang berbeda dalam paket yang sama, dan dari kelas turunan (subclass) bahkan jika subclass berada di luar paket.
* default (tidak menggunakan modifier): Jika tidak ada modifier yang ditentukan, maka member tersebut akan memiliki akses default, yang artinya member tersebut dapat diakses hanya oleh kelas yang berada dalam paket yang sama.

1. Kata kunci super digunakan untuk merujuk ke kelas induk (superclass) dari suatu kelas. Kegunaan utama dari kata kunci super adalah:

Memanggil Konstruktor Kelas Induk: super() digunakan dalam konstruktor kelas turunan untuk memanggil konstruktor dari kelas induk. Hal ini memungkinkan inisialisasi atribut yang diwariskan dari kelas induk.

Mengakses Member Kelas Induk: super digunakan untuk mengakses member (atribut atau metode) dari kelas induk saat ada nama yang sama antara kelas induk dan kelas turunan. Dengan menggunakan super, kita bisa membedakan antara atribut atau metode dari kelas turunan dan kelas induk.

1. Konstruktor tidak diwariskan berarti bahwa konstruktor dari kelas induk tidak secara otomatis diturunkan kepada kelas turunannya. Meskipun kelas turunan akan menggunakan konstruktor dari kelas induk untuk menginisialisasi atribut yang diwarisi, namun konstruktor itu sendiri tidak diturunkan secara langsung.Sebagai contoh:

class Base{

Base(){

int i = 100;

System.out.println(i);

}

}

public class Pri extends Base{

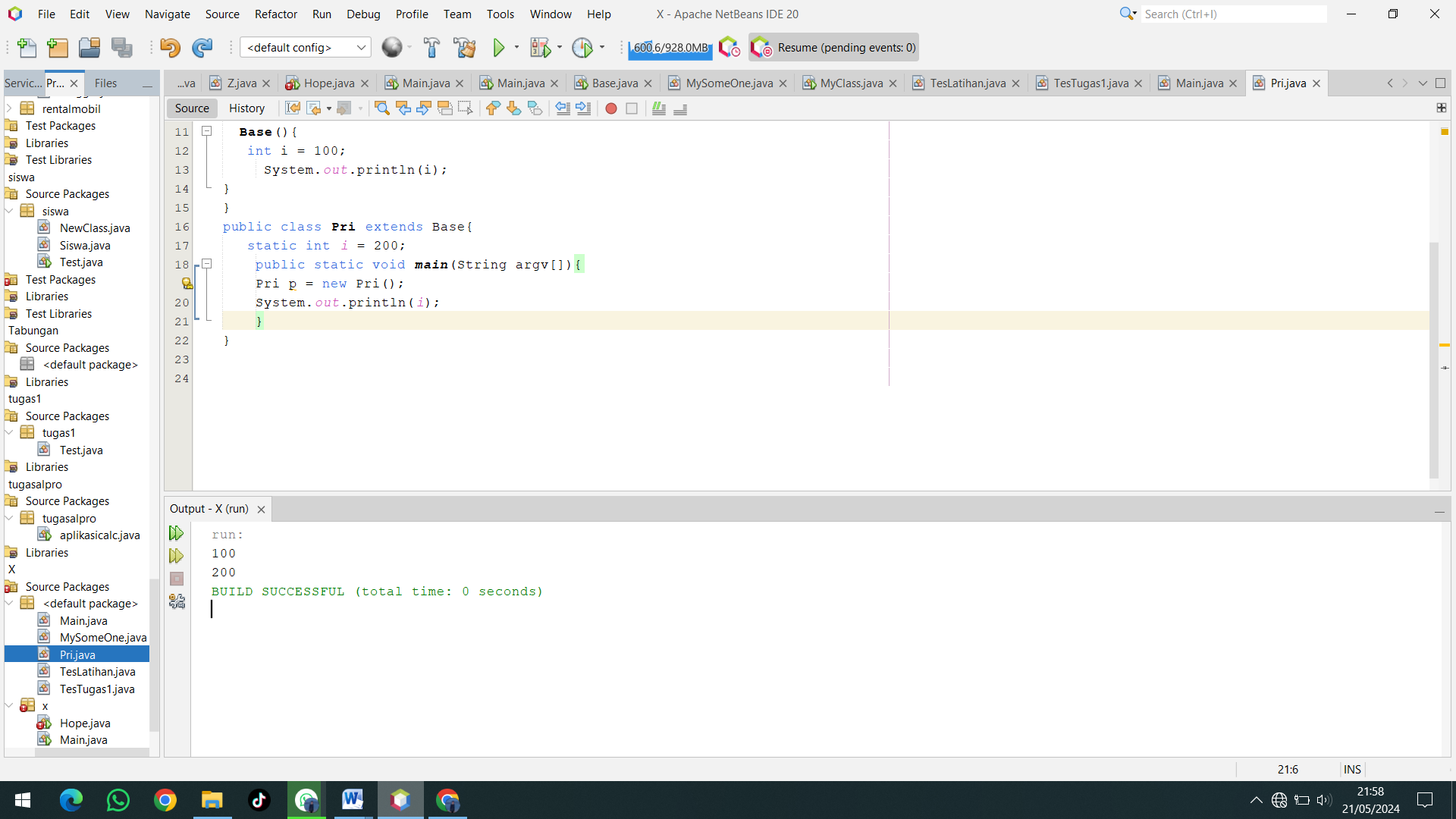
static int i = 200;

public static void main(String argv[]){

Pri p = new Pri();

System.out.println(i);

}



**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(){

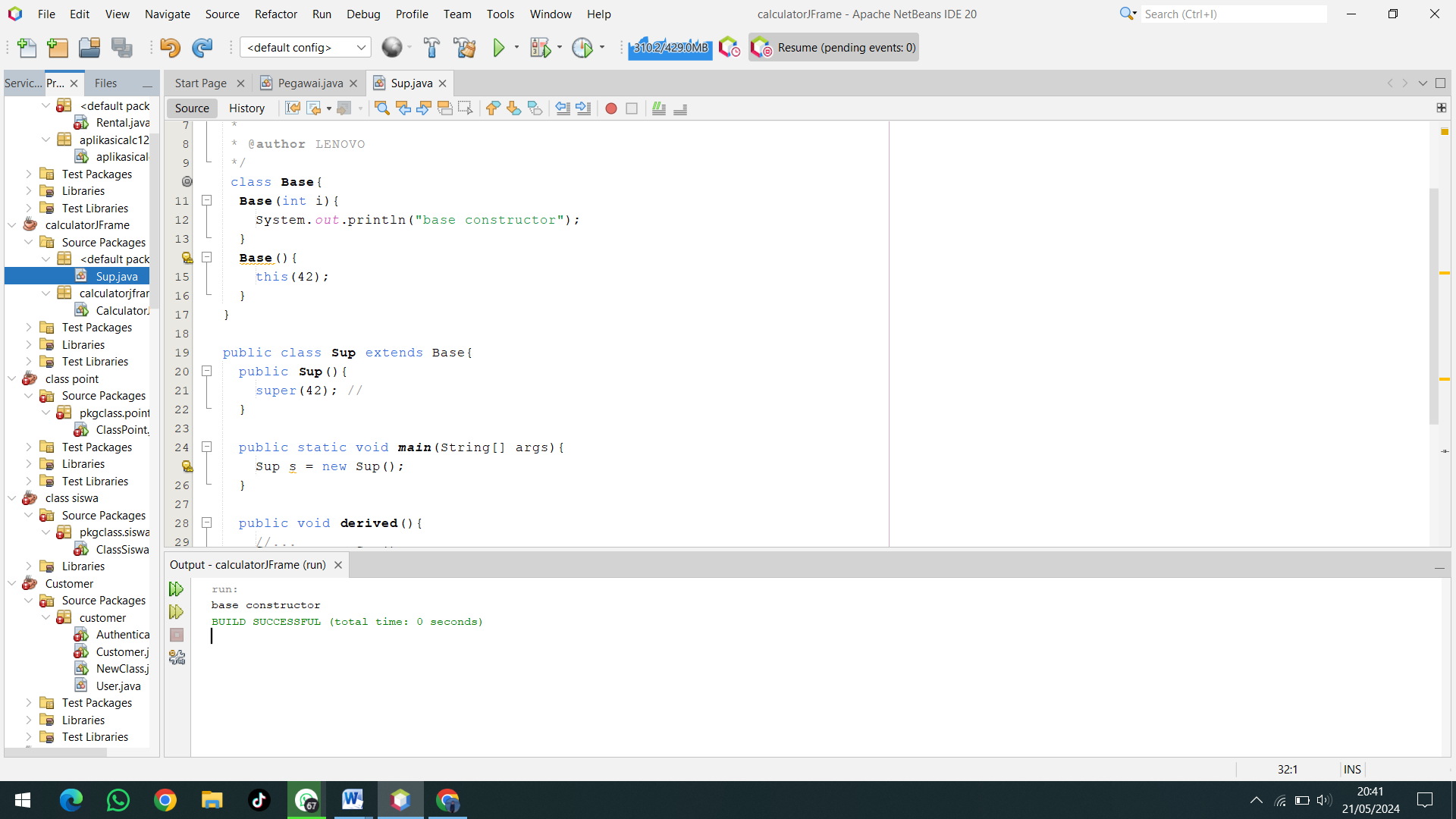
super(42); //

}

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:

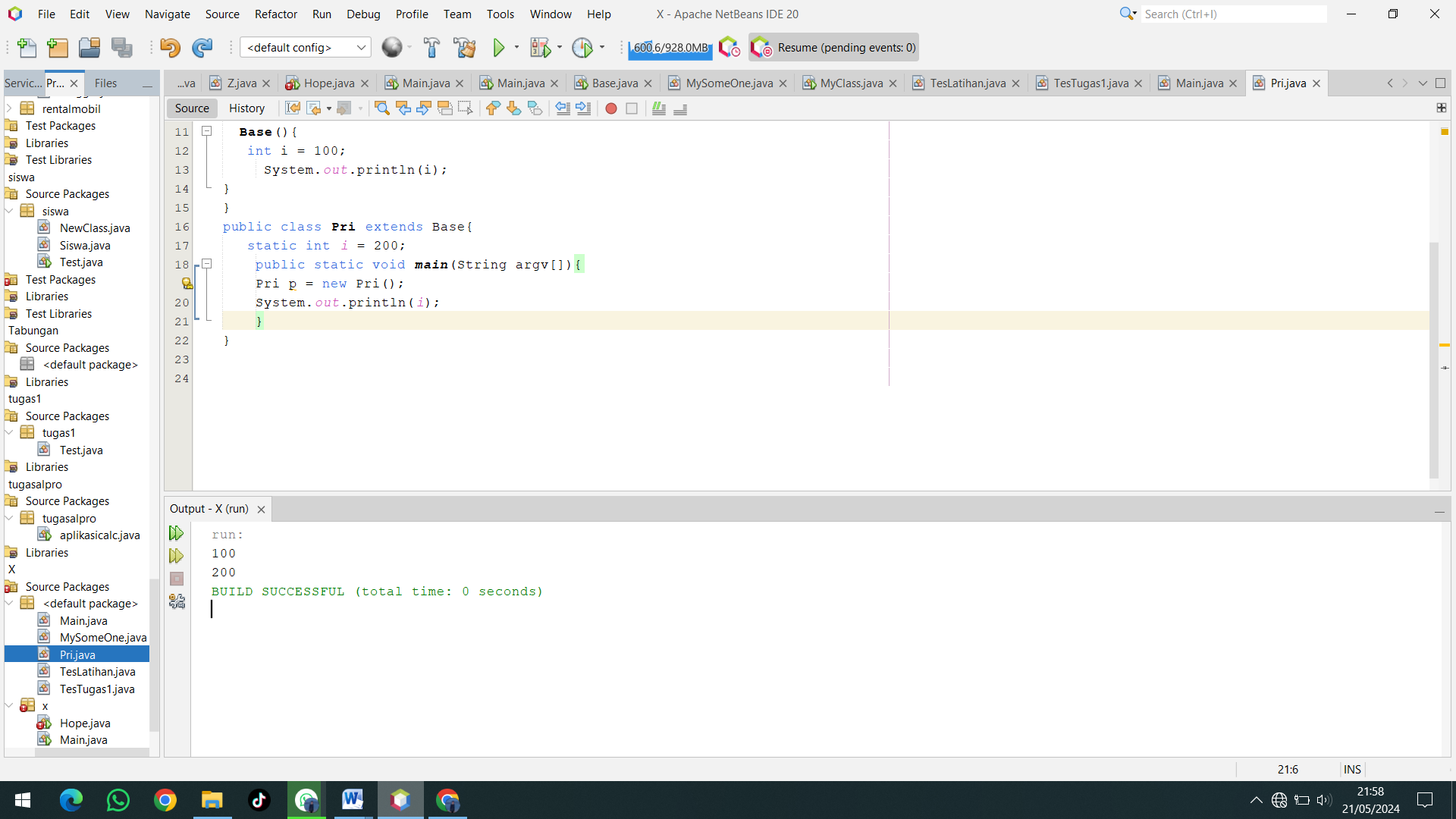
Sebuah variabel statis i dengan nilai 200.

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();

}

}

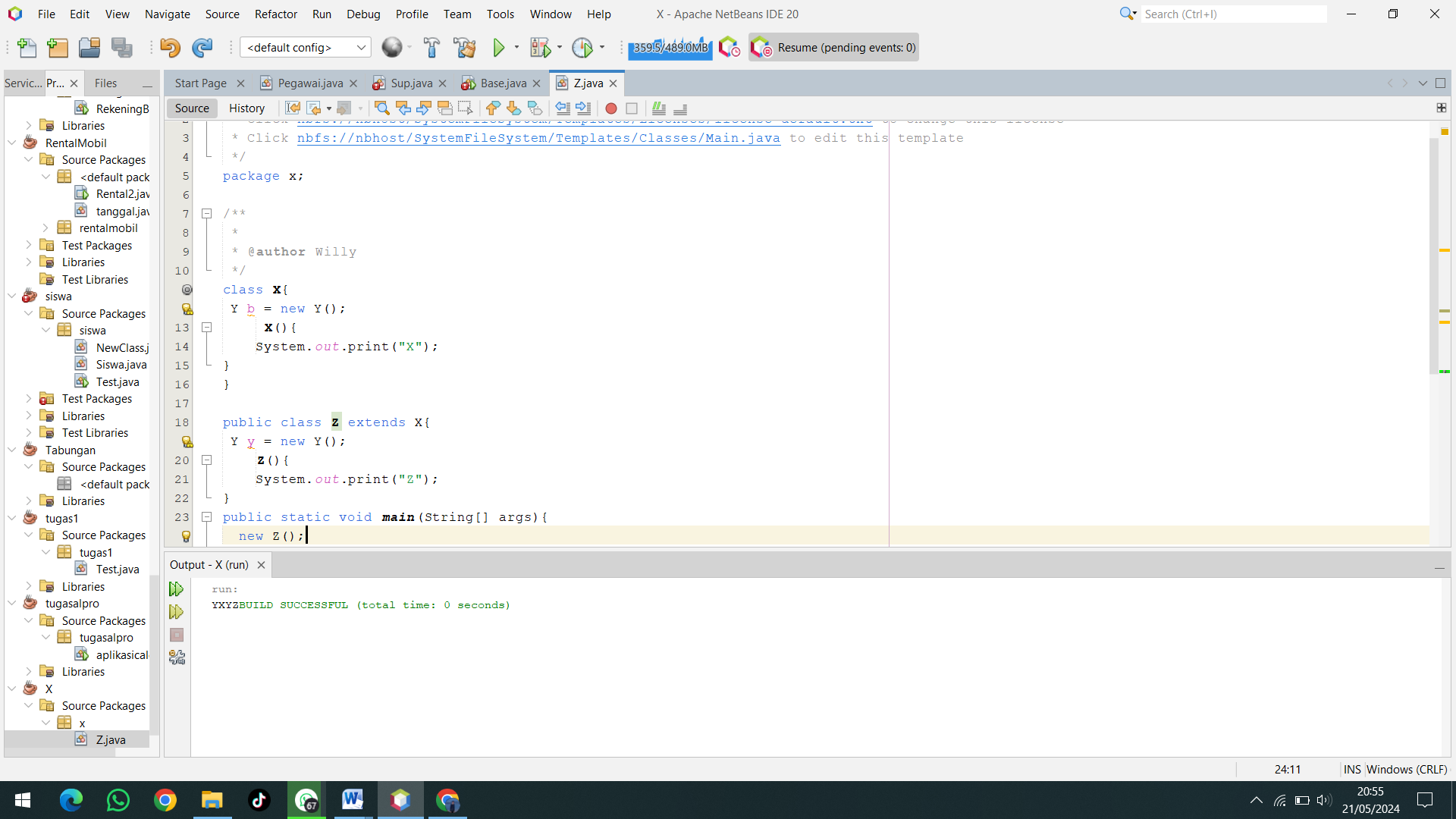
class Y{

Y(){

System.out.print("Y");

}

}



**Latihan 4**

/\*\*

\*

\* @author Willy

\*/

public class Hope{

public static void main(String argv[]){

Hope h = new Hope();

}

protected Hope(){

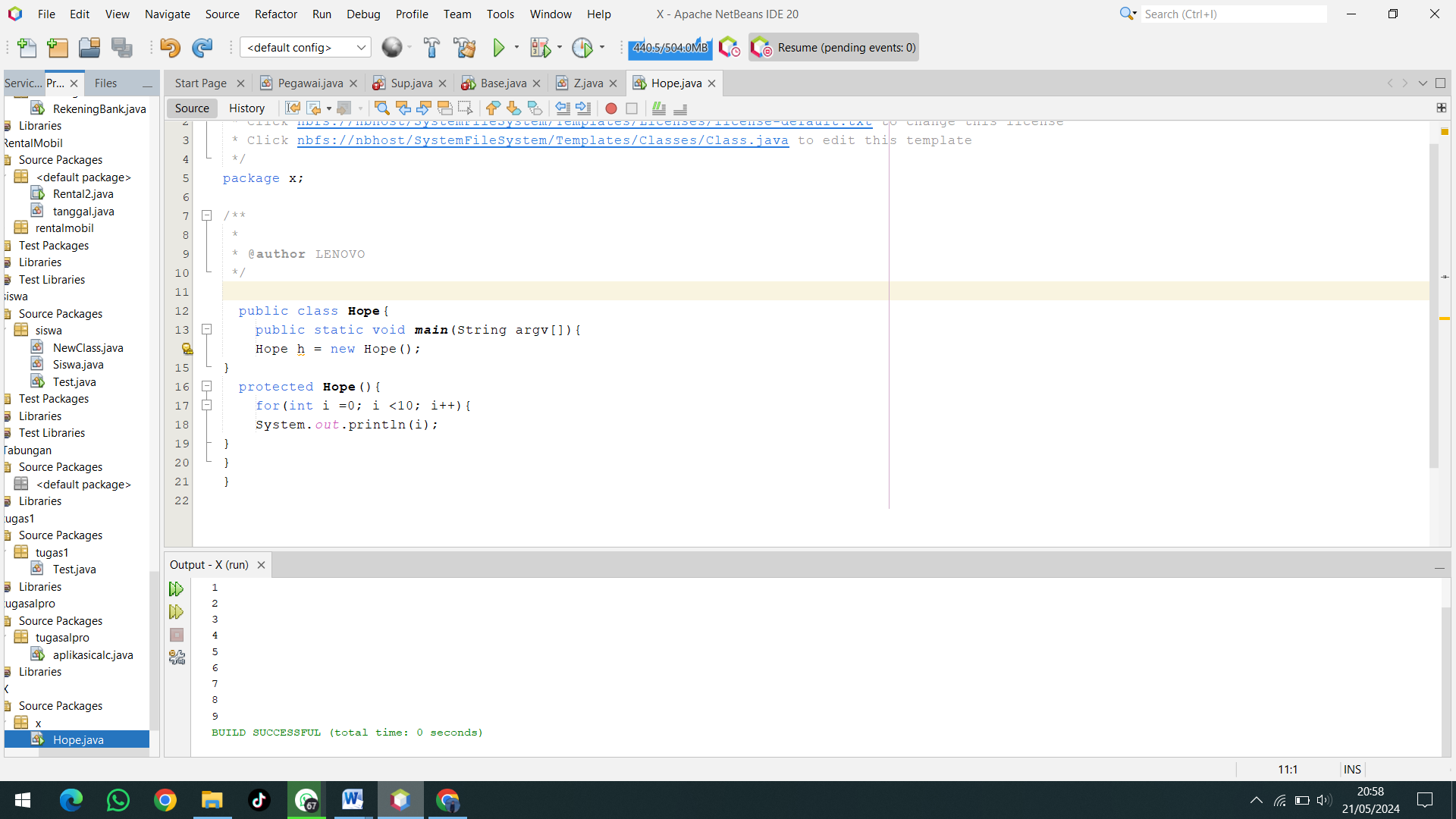
for(int i =0; i <10; i++){

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 :**

