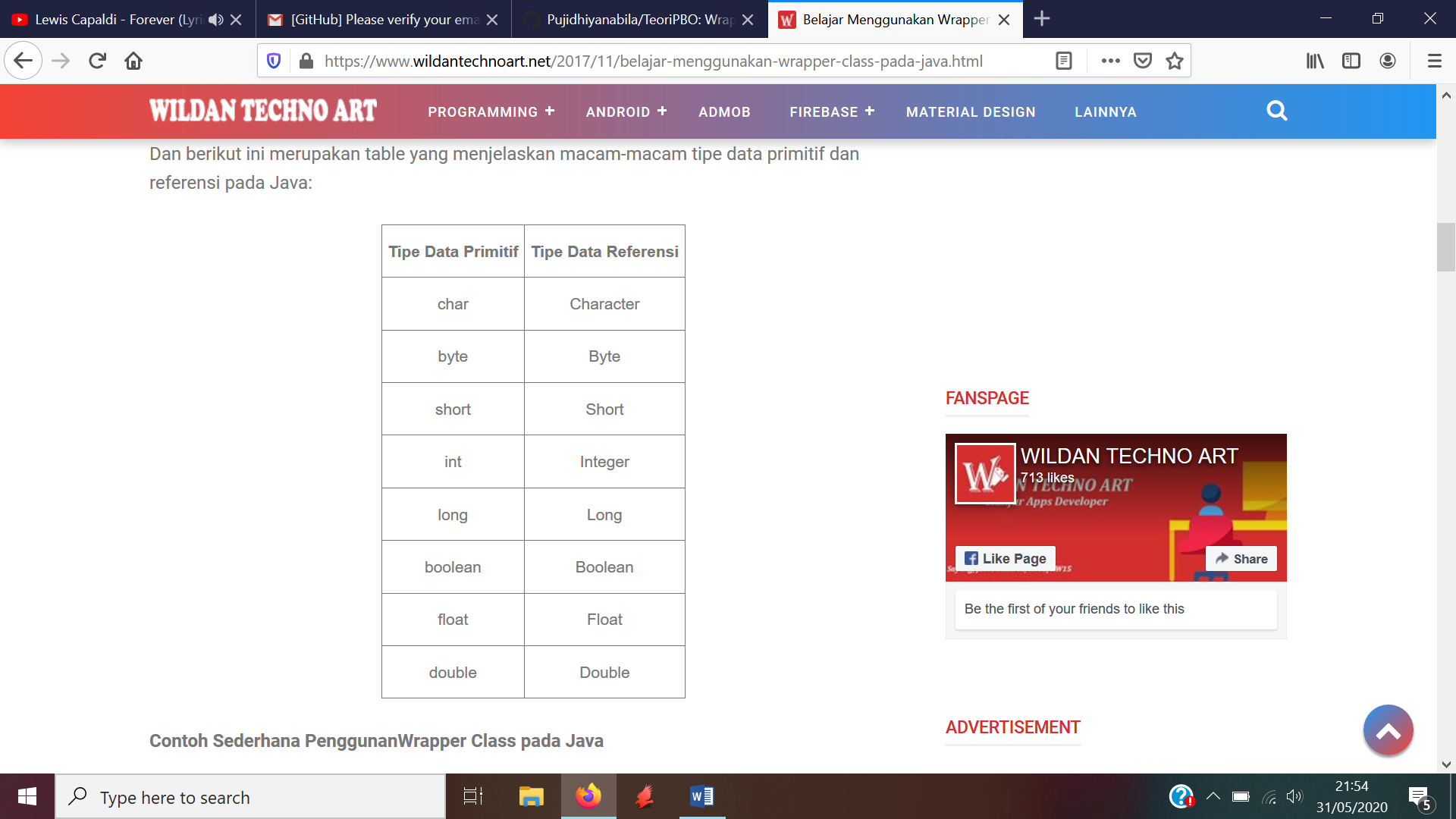
**Wrapper Class pada Java**

***Wrapper Class*** (Kelas Pembungkus) adalah suatu mekanisme yang digunakan untuk mengconvert atau mengubah suatu nilai yang di definisikan, dari tipe data **primitif** menjadi sebuah nilai dengan tipe data **referensi** (Objek). Selain itu, Wrapper Class mendukung method dari tipe data primitif, contohnya seperti **Boolean, Character, Integer, Long, Float, dsb**.

Pada bahasa pemrograman Java, terdapat 2 jenis tipe data, **yaitu primitif dan referensi (objek)**, Tipe Data *Primitif* adalah tipe data yang tidak memiliki method, hanya memiliki data saja. Tipe data ini bukanlah sebuah object, sedangkan Tipe Data *Referensi* adalah tipe data yang digunakan untuk memegang referensi dari sebuah object ( instance dari class).

Dan berikut ini merupakan table yang menjelaskan macam-macam tipe data primitif dan referensi pada Java:



**Contoh Sederhana PenggunanWrapper Class pada Java**  
  
Pada contoh program berikut ini, kita menggunakan Wrapper Class serta mambuat objek dari class tersebut, dengan tipe data, seperti **Character,, Integer, Long, Boolean Float dan Double**. Lalu mendefinisikan nilainya pada argumen didalam Constructornya.

package WrapperClass;

public class WrapperClassExample{

public static void main(String[] args){

/\* Menginstansi Objek Dari Wrapper Class Dan

\* Menggunakan Argumen Dari Tipe Data Primitif

\*/

Character objChar = new Character('P');

Byte objByte = new Byte("12");

Short objShort = new Short("100");

Integer objInt = new Integer(523);

Long objLong = new Long(7689);

Boolean objBoolean = new Boolean(true);

Float objFloat = new Float(82.89f);

Double objDouble = new Double(34.67);

//Menampilkan Output dari Class Wrapper

System.out.println("Character: "+objChar);

System.out.println("Byte: "+objByte);

System.out.println("Short: "+objShort);

System.out.println("Integer: "+objInt);

System.out.println("Long: "+objLong);

System.out.println("Boolean: "+objBoolean);

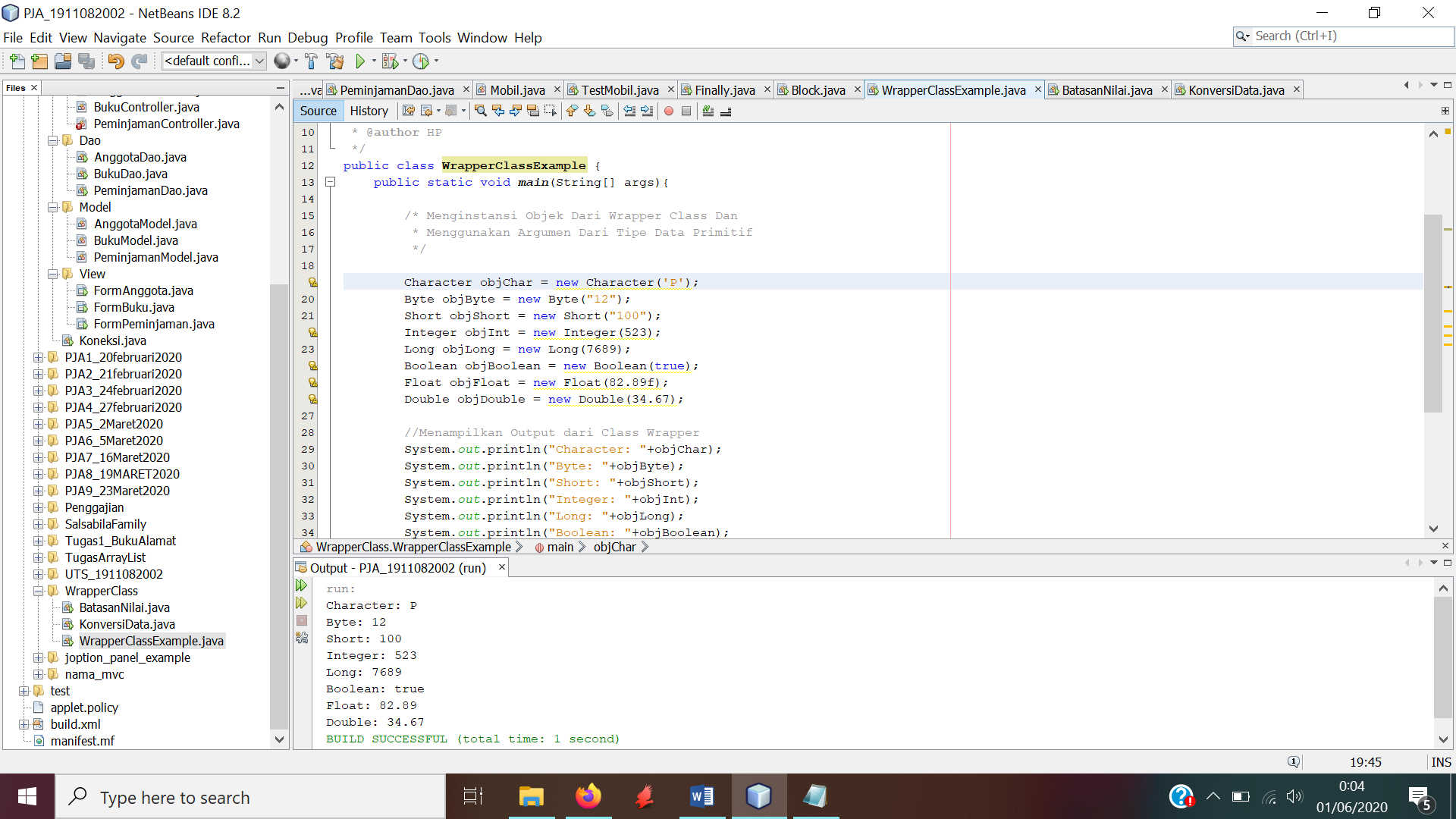
System.out.println("Float: "+objFloat);

System.out.println("Double: "+objDouble);

}

}

**Demo:**



Sedikit penjelasan, pada tipe data Byte dan Short, kalian harus memasukan argumen berupa String, selain menggunakan constructor dengan kata kunci *new*, kalian juga dapat membuat objek serta argumennya seperti berikut ini:

Character objChar = 'P';

Integer objInt = 523;

Boolean objBoolean = true;

Float objFloat = 82.89f;

Double objDouble = 34.67;

Kecuali pada Byte, Short dan Long.  
  
Selain menampilkan nilai dari argumen tersebut, kita juga dapat melihat batasan nilai maximal dan minimal dari tipe data Byte, Short, Integer, Long, Float, Double. Menggunakan fungsi **MAX\_VALUE** dan **MIN\_VALUE**, seperti berikut ini:

public static void main(String[] args){

System.out.println("=============== MAXIMUM VALUE ==============");

//Melihat Batas Maximal Pada Tipe Data Angka (Bilangan Bulat & Floating)

System.out.println("Max Value Pada Byte: "+Byte.MAX\_VALUE);

System.out.println("Max Value Pada Short: "+Short.MAX\_VALUE);

System.out.println("Max Value Pada Integer: "+Integer.MAX\_VALUE);

System.out.println("Max Value Pada Long: "+Long.MAX\_VALUE);

System.out.println("Max Value Pada Float: "+Float.MAX\_VALUE);

System.out.println("Max Value Pada Double: "+Double.MAX\_VALUE);

System.out.println("=============== MINIMUM VALUE ==============");

//Melihat Batas Minimal Pada Tipe Data Angka (Bilangan Bulat & Floating)

System.out.println("Max Value Pada Byte: "+Byte.MIN\_VALUE);

System.out.println("Max Value Pada Short: "+Short.MIN\_VALUE);

System.out.println("Max Value Pada Integer: "+Integer.MIN\_VALUE);

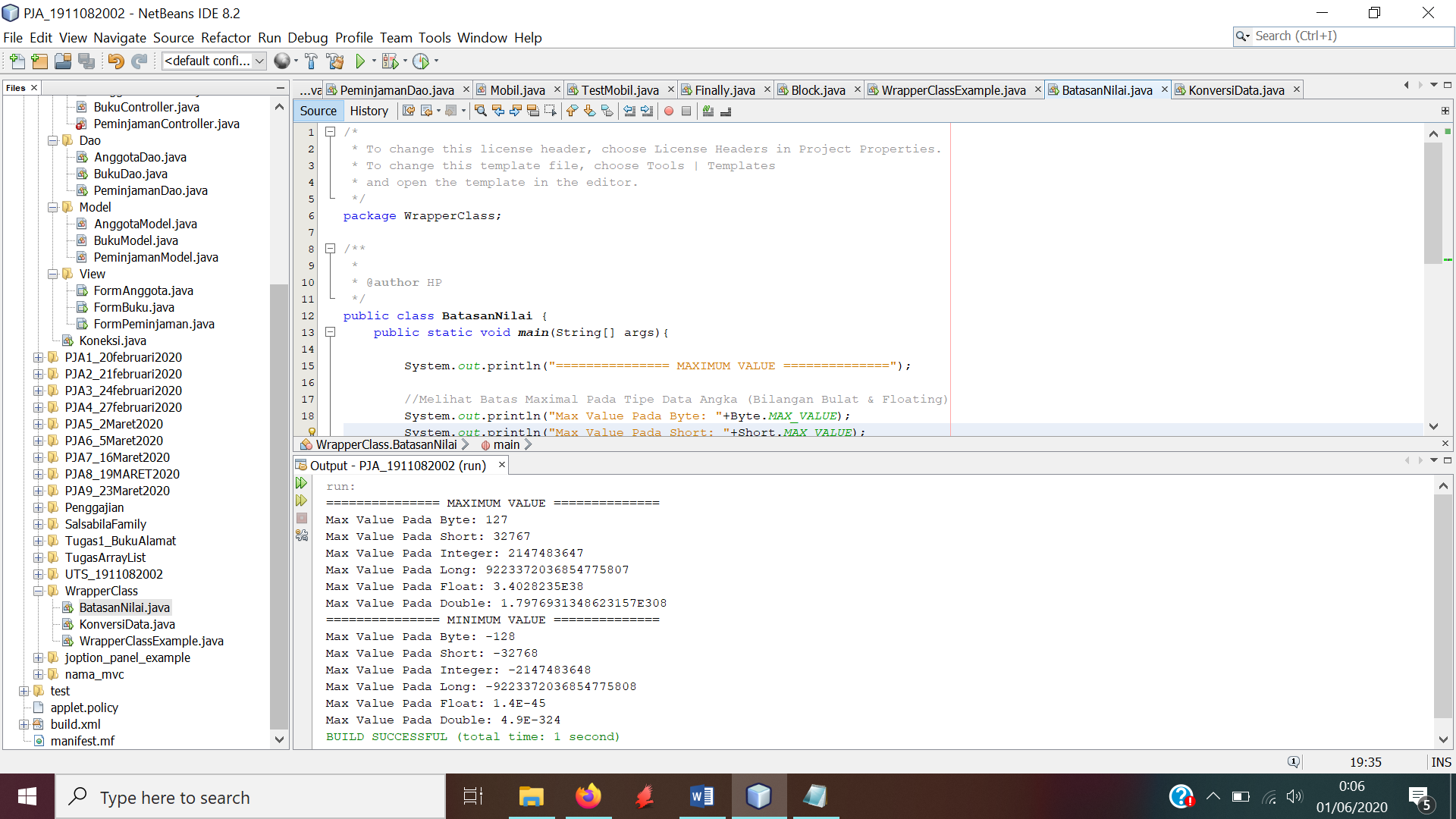
System.out.println("Max Value Pada Long: "+Long.MIN\_VALUE);

System.out.println("Max Value Pada Float: "+Float.MIN\_VALUE);

System.out.println("Max Value Pada Double: "+Double.MIN\_VALUE);

}

**Demo:**



**Konversi Tipe Data Primitif ke Wrapper Class (Referensi), dan sebaliknya:**   
  
Pada contoh program selanjutnya, kita akan mengubah atau mengkonvert tipe data Primitif ke Referensi atau sebaliknya, seperti char ke Character, int ke Integer, boolean ke Boolean, double ke Double dan Sebaliknya:

public static void main(String[] args){

//Mengkonversi Tipe Data Primitif ke Wrapper Class (Referensi)

char karakter = 'P';

int angka = 256;

boolean status = true;

double pecahan = 67.3;

Character ConvChar = Character.valueOf(karakter); //Konversi char ke Charcter

Integer ConvInt = Integer.valueOf(angka); //Konversi int ke Integer

Boolean ConvBoolaen = Boolean.valueOf(status); //Konversi boolean ke Boolean

Double ConvDouble = Double.valueOf(pecahan); //Konversi double ke Double

//Mengkonversi Tipe Data Wrapper Class (Referensi) ke Primitif

Character refChar = new Character('P');

Integer refInt = new Integer(332);

Boolean refBoolean = new Boolean(true);

Double refDouble = new Double(84.63);

char primChar = refChar.charValue(); //Konversi Charcter ke char

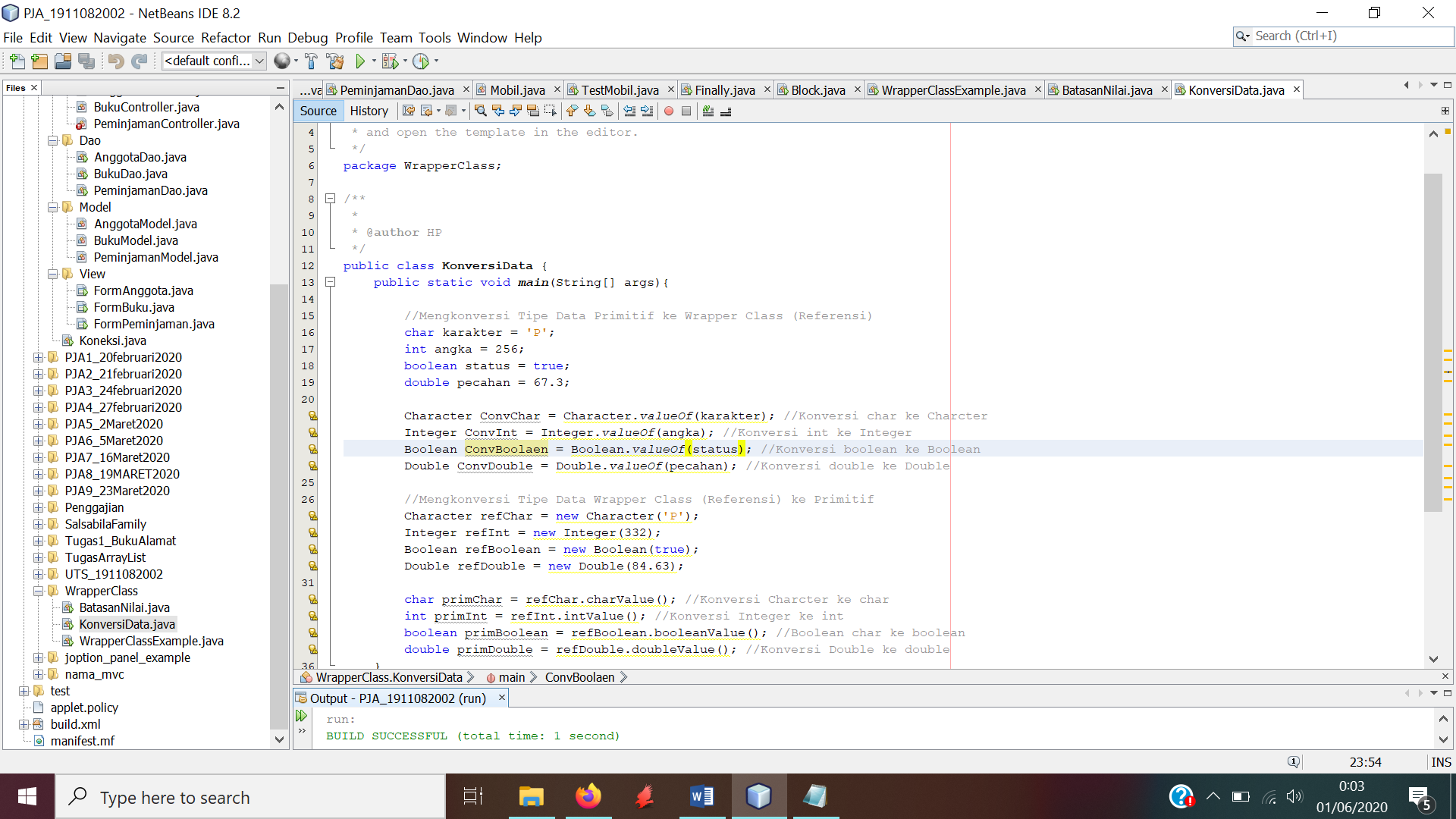
int primInt = refInt.intValue(); //Konversi Integer ke int

boolean primBoolean = refBoolean.booleanValue(); //Boolean char ke boolean

double primDouble = refDouble.doubleValue(); //Konversi Double ke double

}

**Demo:**



Pada contoh program tersebut, untuk mengubah tipe data Primitif ke Wrapper Class, kalian dapat menggunakan fungsi **valueOf()** dan untuk sebaliknya, kalian dapat menggunakan fungsi **charValue, intValue, booleanValue, dsb**. Tergantung dengan tipe data yang kalian gunakan

***Perbedaan Final, Finally dan Finalize***

***1. Final*** adalah keyword untuk mendeklarasikan bahwa sebuah variabel/atribut tidak dapat diubah lagi isinya (konstan), jika final disebutkan di method, artinya method tersebut tidak dapat di override oleh subclassnya, jika final disebutkan di class, artinya class tersebut tidak dapat diturunkan lagi

public final class Mobil {

    private String nama;

    public String getNama() {

        return nama;

    }

    public void setNama(String nama) {

        this.nama = nama;

    }

}

**Main Methodnya :**

public class TestMobil {

public static void main(String args []){

Mobil mobil = new Mobil();

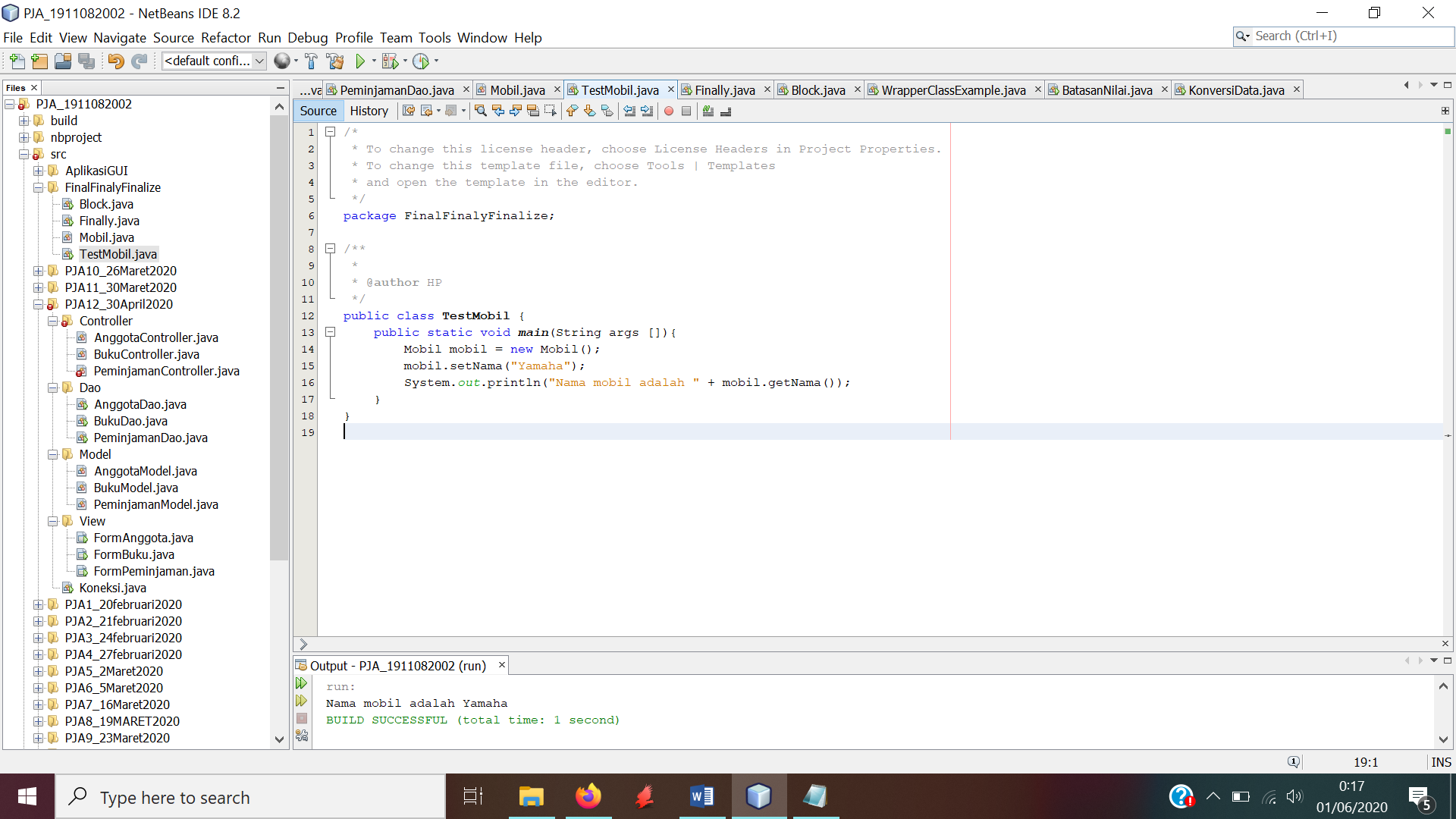
mobil.setNama("Yamaha");

System.out.println("Nama mobil adalah " + mobil.getNama());

}

}

**Output :**



***Finally*** merupakan tag dalam try-finally yang digunakan untuk memastikan bahwa kode didalam finally akan dieksekussi walaupun dalam try sukses / gagal.

public class Finally {

public static void main(String[] args) {

int i = 0;

String myarray[] = {"Array pertama","Array kedua","Array ketiga"};

while (i < 4) {

try {

System.out.println(myarray[i]);

i++;

} catch (ArrayIndexOutOfBoundsException e){

System.out.println("Re-setting Index Value");

i = 0;

} finally {

System.out.println("This is always printed");

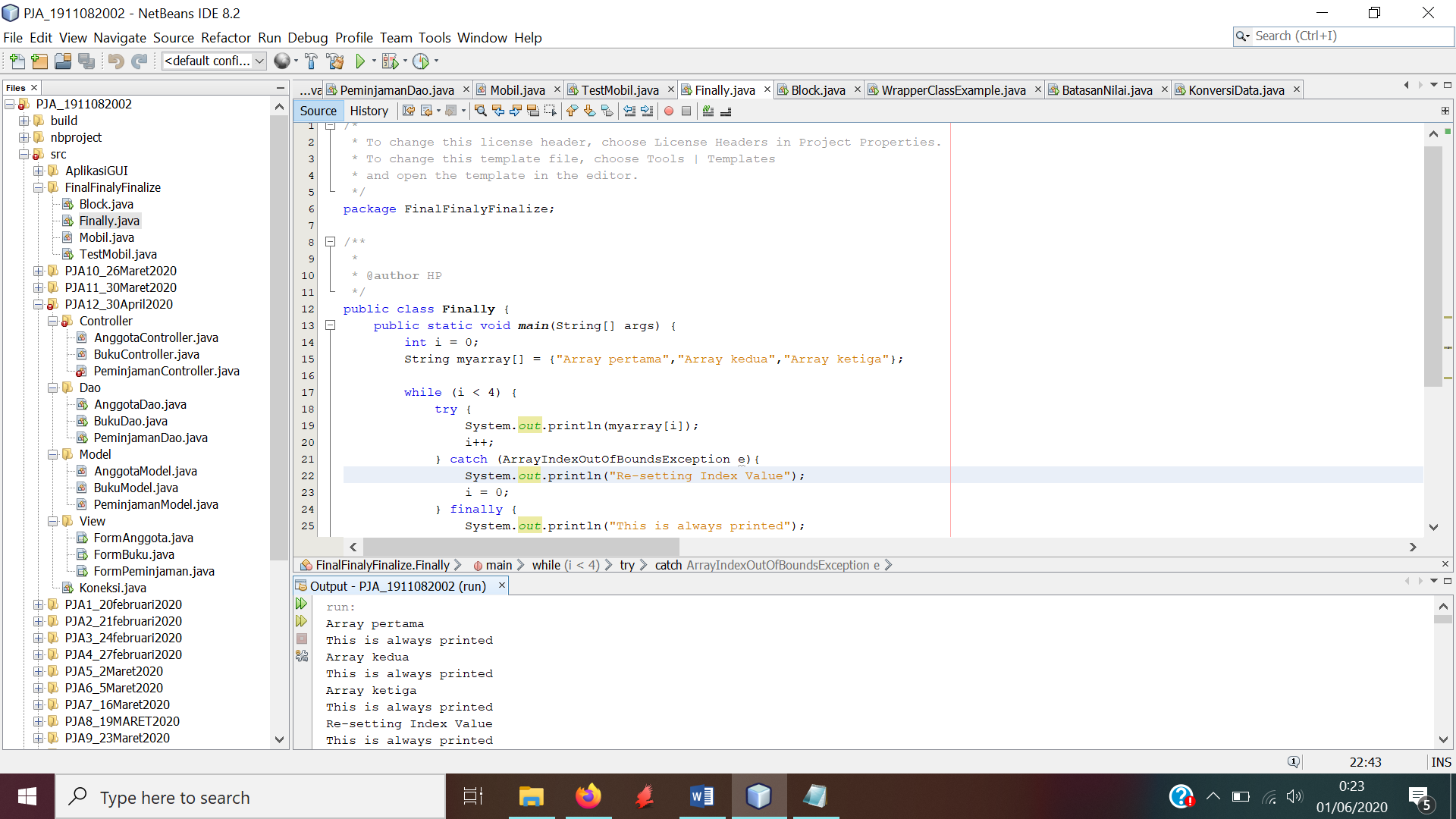
}

}

}

}

**Output :**



***Finalize*** merupakan methode kelas Object yang dieksekusi saat garbace collection menghapus sebuah object yang sudah tidak terpakai.

package FinalFinalyFinalize;

public class Block{

//Method ini akan ditampilkan jika Garbage Collection bekerja

public void finalize(){

System.out.println("Objek Yang Tidak Terpakai Sudah Dibersihkan:");

}

public static void main(String[] args){

Runtime RT = Runtime.getRuntime();

System.out.println("Jumlah Memori Awal: "+RT.totalMemory());

Block objek1 = new Block(); //Block Satu

Block objek2 = new Block(); //Block Dua

Block objek3 = objek1; //Objek3 Mengembalikan Nilai Objek1

objek1 = null;

objek2 = null;

System.out.println("Jumlah Memori Yeng Tersedia Sebelum di GC: "+RT.freeMemory());

System.gc();

System.out.println("===========================================");

System.out.println("Jumlah Memori Yeng Tersedia Setelah di GC: "+RT.freeMemory());

}

}

**Output :**

