**TUGAS 12**

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



**Oleh**

**WILLY ACHMAD NURANI**

**(4123024)**

**PROGAM-STRATA**

**JURUSAN SISTEM INFORMASI**

**UNIVERSITAS PESANTREN TINGGI DARUL 'ULUM JOMBANG**

**2024**

1. **Percobaan**

* **Percobaan 1**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.\*;

public class SetExample {

public static void main(String[] args) {

Set set=new HashSet();

set.add("Bernadine");

set.add("Elizabeth");

set.add("Gene");

set.add("Elizabeth");

set.add("Clara");

System.out.print("Elemen pada HashSet : ");

System.out.println(set);

Set sortSet=new TreeSet(set);

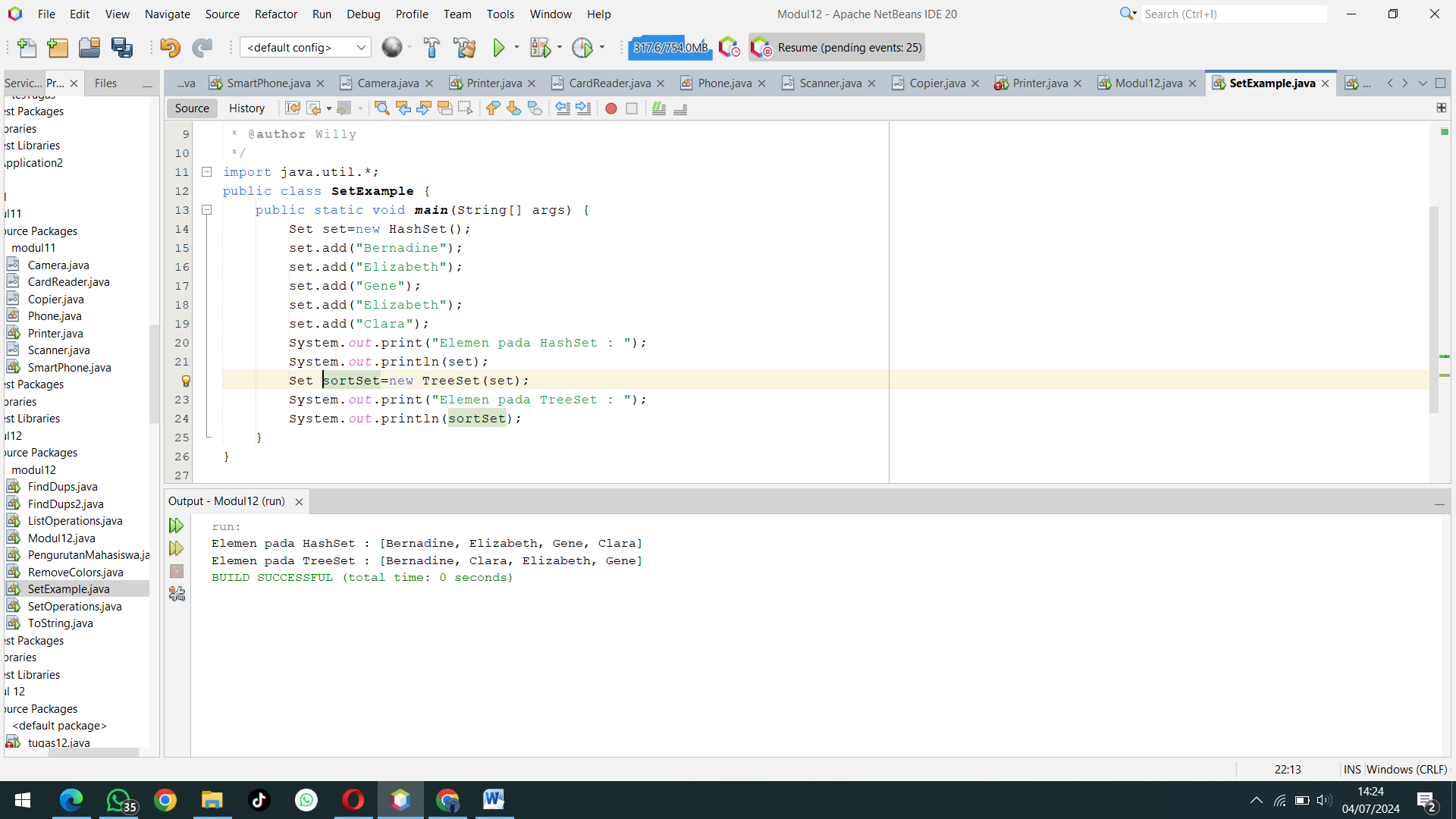
System.out.print("Elemen pada TreeSet : ");

System.out.println(sortSet);

}

}

**Output :**



1. Paket dan Impor

* Kode berada di dalam paket modul12.
* Mengimpor semua kelas dari java.util untuk menggunakan koleksi seperti Set, HashSet, dan TreeSet.

1. Deklarasi Kelas

* Kelas SetExample dideklarasikan sebagai kelas publik dengan metode main.

1. Metode main

* Membuat objek HashSet dan menyimpannya dalam variabel set.
* Menambahkan elemen-elemen ke HashSet:
* "Bernadine"
* "Elizabeth"
* "Gene"
* "Elizabeth" (duplikat, tidak akan ditambahkan karena HashSet tidak menyimpan duplikat)
* "Clara"
* Menampilkan elemen-elemen dalam HashSet:
* Output tidak dapat diprediksi dalam urutan tertentu karena HashSet tidak menjamin urutan.

1. Pembuatan dan Penggunaan TreeSet

* Membuat objek TreeSet dengan set sebagai argumen konstruktornya. TreeSet secara otomatis mengurutkan elemen-elemennya.
* Menampilkan elemen-elemen dalam TreeSet:
* Output akan diurutkan secara alami (berdasarkan urutan abjad untuk string).
* **Percobaan 2**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.HashSet;

import java.util.Set;

public class FindDups {

public static void main(String[] args) {

Set<String> s = new HashSet<String>();

for (String a : args) {

if (!s.add(a)) {

System.out.println("Duplicate detected: " + a);

}

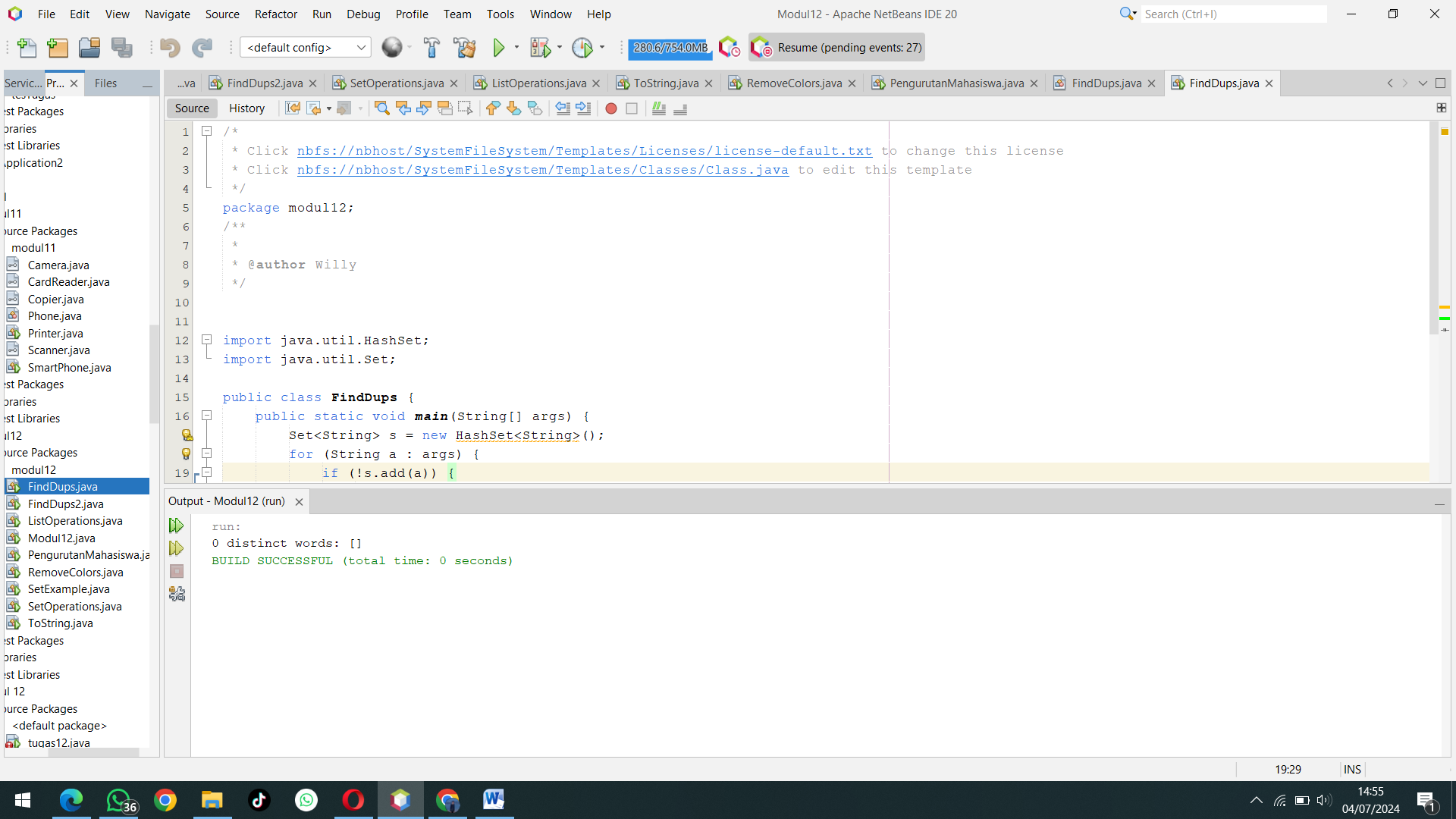
}

System.out.println(s.size() + " distinct words: " + s);

}

}

**Output :**



1. Deskripsi Kode

Kode ini bertujuan untuk mengidentifikasi dan menampilkan duplikat dari argumen yang diberikan dalam bentuk string. Berikut adalah rincian dari kode:

* 1. Penggunaan Set dan HashSet
* Set adalah antarmuka di Java yang tidak mengizinkan elemen duplikat. Implementasi umum dari Set adalah HashSet, yang digunakan di sini.
* HashSet menyimpan elemen secara acak tanpa urutan khusus dan tidak mengizinkan elemen duplikat.

1. Logika Kode

* Program membaca argumen yang diberikan pada baris perintah (args), dan untuk setiap elemen, mencoba menambahkannya ke dalam HashSet.
* Metode add() dari HashSet mengembalikan false jika elemen yang ingin ditambahkan sudah ada dalam set, yang berarti ada duplikat.
* Jika add() mengembalikan false, maka elemen tersebut adalah duplikat, dan program akan mencetak pesan yang menunjukkan elemen duplikat tersebut.
* Setelah memeriksa semua elemen, program mencetak jumlah elemen unik dan isi dari HashSet.

1. Analisis Kode
2. Inisialisasi HashSet

* Set<String> s = new HashSet<String>();:
* Membuat instance HashSet untuk menyimpan string unik yang dibaca dari argumen.

1. Iterasi dan Pemeriksaan Duplikat

* for (String a : args):
  + Iterasi melalui setiap elemen yang diterima sebagai argumen.
* if (!s.add(a)):
  + Menambahkan elemen ke dalam HashSet. Jika elemen sudah ada di set, add() mengembalikan false, menandakan bahwa elemen tersebut adalah duplikat.
* System.out.println("Duplicate detected: " + a);:
  + Mencetak elemen yang sudah ada di set, menandakan bahwa itu adalah duplikat.

1. Menampilkan Hasil Akhir

* System.out.println(s.size() + " distinct words: " + s);:
  + Mencetak jumlah elemen unik (ukuran HashSet) dan konten dari HashSet.
* **Percobaan 3**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.\*;

public class SetExample {

public static void main(String[] args) {

Set s1=new HashSet();

s1.add("Australia");

s1.add("Sweden");

s1.add("Germany");

Set s2=new HashSet();

s2.add("Sweden");

s2.add("France");

Set union=new TreeSet(s1);

union.addAll(s2); // gabungan dari s1 dan s2

print("Union",union);

Set intersect=new TreeSet(s1);

intersect.retainAll(s2); // irisan dari s1 dan s2

print("Intersection",intersect);

}

protected static void print(String label, Collection c){

System.out.println("---------------- "+ label+" ---------------");

Iterator it=c.iterator();

while(it.hasNext()){

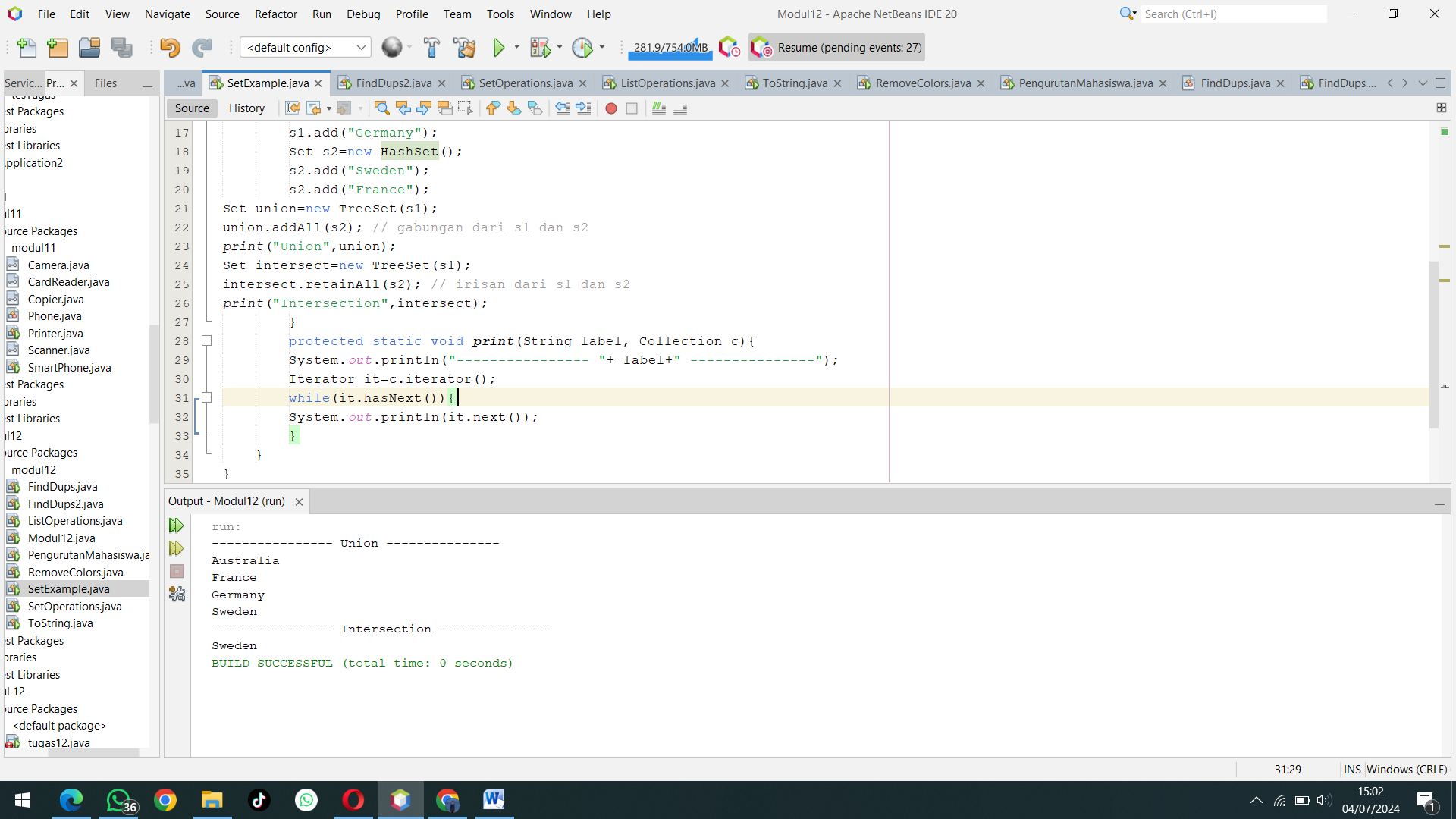
System.out.println(it.next());

}

}

}

**Output :**



1. Deskripsi Kode

Kode ini bertujuan untuk menunjukkan penggunaan Set dalam Java, termasuk operasi gabungan (union) dan irisan (intersection) antara dua set. Kode ini menggunakan HashSet untuk menyimpan elemen dan TreeSet untuk melakukan operasi gabungan dan irisan.

1. Analisis Kode
2. Deklarasi dan Inisialisasi Set

* Set s1 = new HashSet();
  + Membuat sebuah HashSet bernama s1 dan menambahkan tiga elemen: "Australia", "Sweden", dan "Germany". HashSet tidak mempertahankan urutan elemen.
* Set s2 = new HashSet();
  + Membuat sebuah HashSet bernama s2 dan menambahkan dua elemen: "Sweden" dan "France".

1. Operasi Union (Gabungan)

* Set union = new TreeSet(s1);
  + Membuat sebuah TreeSet bernama union dan menginisialisasinya dengan elemen-elemen dari s1. TreeSet secara otomatis mengurutkan elemen-elemen yang dimasukkan.
* union.addAll(s2);
  + Menambahkan semua elemen dari s2 ke union. Metode addAll() akan menambahkan elemen dari s2 ke union jika elemen tersebut belum ada di union, sehingga menghasilkan gabungan dari s1 dan s2.

1. Operasi Intersection (Irisan)

* Set intersect = new TreeSet(s1);
  + Membuat sebuah TreeSet bernama intersect dan menginisialisasinya dengan elemen-elemen dari s1. Ini akan digunakan untuk menyimpan elemen yang ada di kedua set (s1 dan s2).
* intersect.retainAll(s2);
  + Menyimpan hanya elemen-elemen yang ada di kedua set dalam intersect. Metode retainAll() mengubah intersect menjadi irisan dari s1 dan s2.

1. Metode print

* protected static void print(String label, Collection c)
  + Metode ini digunakan untuk mencetak isi dari koleksi (Collection). Menerima dua parameter: label yang menunjukkan jenis koleksi yang dicetak, dan koleksi itu sendiri.
  + Menggunakan iterator untuk mengakses dan mencetak setiap elemen dari koleksi.
* **Percobaan 4**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.\*;

public class ListExample {

public static void main(String[] args) {

List list=new ArrayList();

list.add("Bernadine");

list.add("Elizabeth");

list.add("Gene");

list.add("Elizabeth");

list.add("Clara");

System.out.println(list);

System.out.println("2 : "+list.get(2));

System.out.println("0 : "+list.get(0));

LinkedList queue=new LinkedList();

queue.addFirst("Bernadine");

queue.addFirst("Elizabeth");

queue.addFirst("Gene");

queue.addFirst("Elizabeth");

queue.addFirst("Clara");

System.out.println(queue);

queue.removeLast();

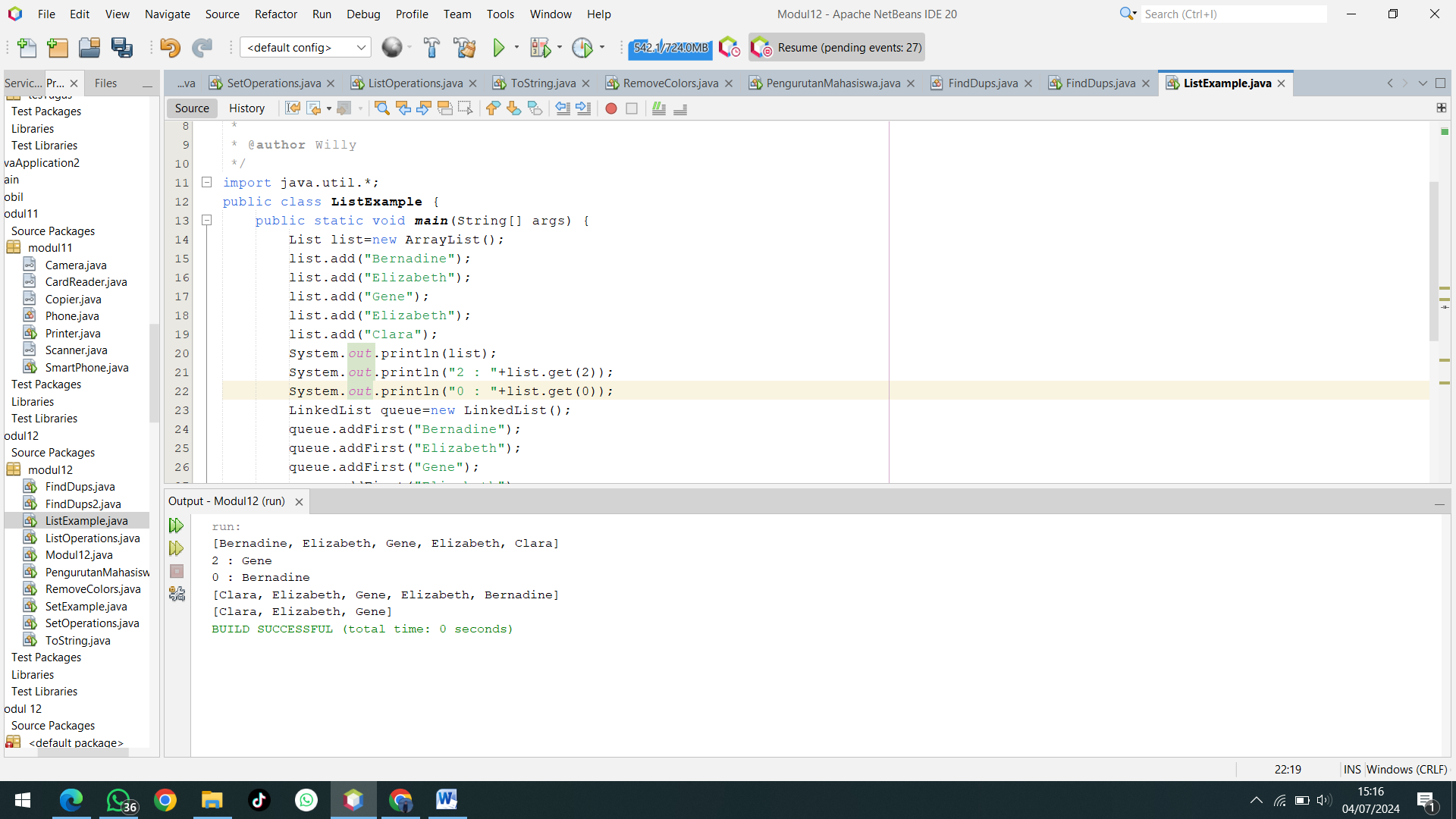
queue.removeLast();

System.out.println(queue);

}

}

**Output:**



1. Deskripsi Kode

Kode ini mendemonstrasikan penggunaan ArrayList dan LinkedList di Java. Kode ini menambahkan elemen ke dalam kedua jenis list, melakukan operasi dasar seperti pengambilan dan penghapusan elemen, serta mencetak hasilnya.

1. Analisis Kode
2. Penggunaan ArrayList

* List list = new ArrayList();
  + Membuat instance ArrayList yang dapat menyimpan elemen. ArrayList adalah implementasi dari List yang menyimpan elemen dalam urutan dan memungkinkan duplikasi.
* list.add("Bernadine");
  + Menambahkan elemen "Bernadine" ke ArrayList.
* list.add("Elizabeth");
  + Menambahkan elemen "Elizabeth" ke ArrayList.
* list.add("Gene");
  + Menambahkan elemen "Gene" ke ArrayList.
* list.add("Elizabeth");
  + Menambahkan elemen "Elizabeth" lagi ke ArrayList (menunjukkan bahwa ArrayList mengizinkan duplikasi).
* list.add("Clara");
  + Menambahkan elemen "Clara" ke ArrayList.
* System.out.println(list);
  + Mencetak semua elemen dalam ArrayList. Outputnya adalah: [Bernadine, Elizabeth, Gene, Elizabeth, Clara].
* System.out.println("2 : " + list.get(2));
  + Mengambil dan mencetak elemen pada indeks ke-2 dari ArrayList, yaitu "Gene".
* System.out.println("0 : " + list.get(0));
  + Mengambil dan mencetak elemen pada indeks ke-0 dari ArrayList, yaitu "Bernadine".

1. Penggunaan LinkedList

* LinkedList queue = new LinkedList();
  + Membuat instance LinkedList. LinkedList juga merupakan implementasi dari List, tetapi lebih efisien untuk operasi yang melibatkan penambahan atau penghapusan elemen di awal atau akhir list dibandingkan dengan ArrayList.
* queue.addFirst("Bernadine");
  + Menambahkan elemen "Bernadine" ke awal LinkedList. addFirst menambahkan elemen di depan list.
* queue.addFirst("Elizabeth");
  + Menambahkan elemen "Elizabeth" ke awal LinkedList. Elemen ini akan berada di depan "Bernadine".
* queue.addFirst("Gene");
  + Menambahkan elemen "Gene" ke awal LinkedList. Elemen ini akan berada di depan "Elizabeth".
* queue.addFirst("Elizabeth");
  + Menambahkan elemen "Elizabeth" ke awal LinkedList. Elemen ini akan berada di depan "Gene".
* queue.addFirst("Clara");
  + Menambahkan elemen "Clara" ke awal LinkedList. Elemen ini akan berada di depan "Elizabeth".
* System.out.println(queue);
  + Mencetak semua elemen dalam LinkedList. Outputnya adalah: [Clara, Elizabeth, Gene, Elizabeth, Bernadine].
* queue.removeLast();
  + Menghapus elemen terakhir dari LinkedList, yaitu "Bernadine".
* queue.removeLast();
  + Menghapus elemen terakhir yang tersisa dari LinkedList, yaitu "Elizabeth".
* System.out.println(queue);
  + Mencetak semua elemen yang tersisa dalam LinkedList. Outputnya adalah: [Clara, Elizabeth, Gene].
* **Percobaan 5**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.Vector;

public class VectorExample {

public static void main(String[] args) {

Vector<String> vc=new Vector<String>();

//<E> Element type of Vector e.g. String, Integer, Object ...

// add vector elements

vc.add("Vector Object 1");

vc.add("Vector Object 2");

vc.add("Vector Object 3");

vc.add("Vector Object 4");

vc.add("Vector Object 5");

// add vector element at index

vc.add(3, "Element at fix position");

// vc.size() inform number of elements in Vector

System.out.println("Vector Size :"+vc.size());

// get elements of Vector

for(int i=0;i<vc.size();i++)

{

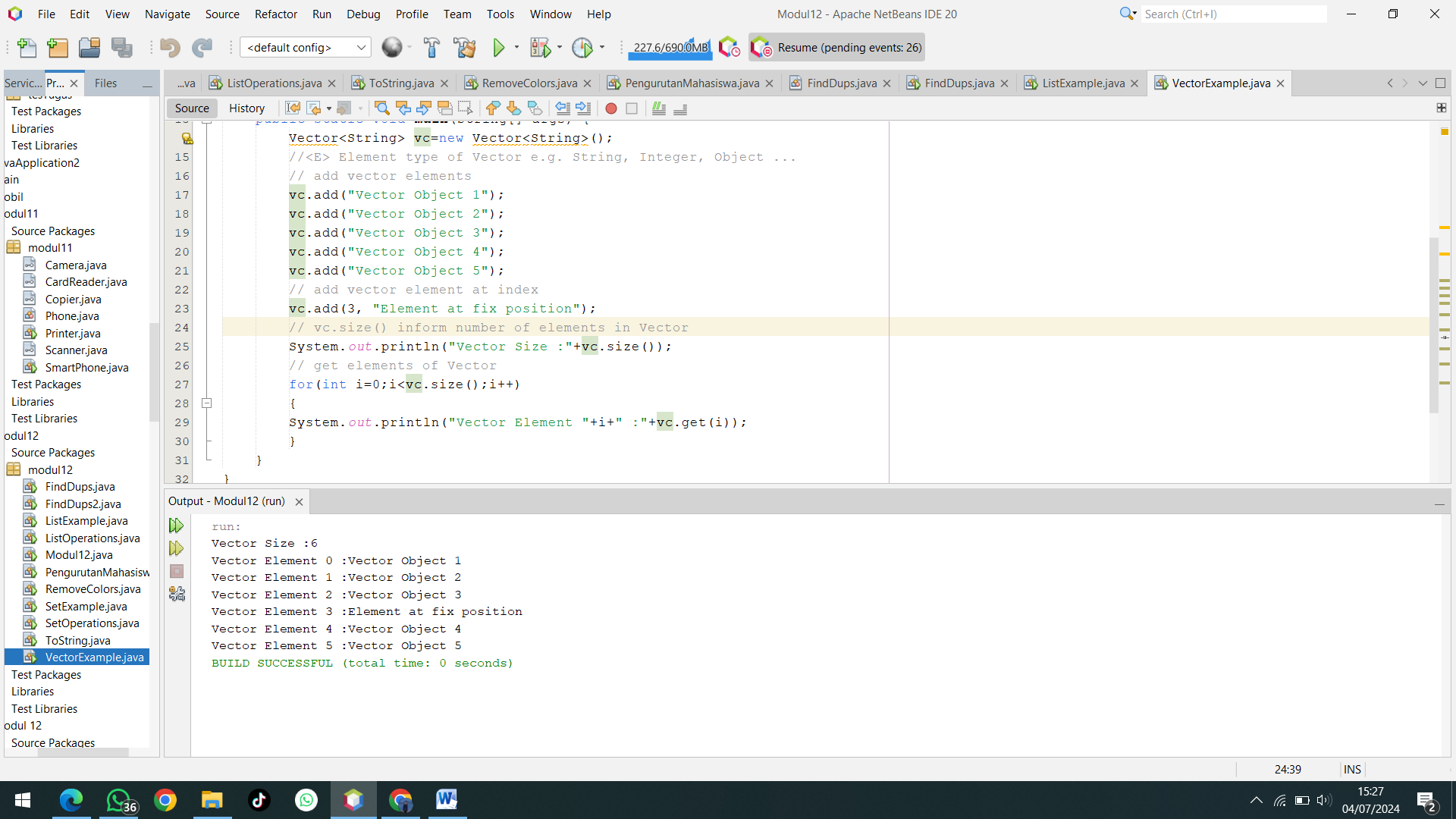
System.out.println("Vector Element "+i+" :"+vc.get(i));

}

}

}

**Output:**



1. Deskripsi Kode

Kode ini mendemonstrasikan penggunaan kelas Vector di Java. Vector adalah implementasi dari struktur data list yang menyimpan elemen dalam urutan dan memungkinkan akses berbasis indeks. Kode ini menambahkan elemen ke dalam Vector, menambahkan elemen pada indeks tertentu, dan mencetak elemen serta ukuran Vector.

1. Analisis Kode
2. Penggunaan Vector

* Vector<String> vc = new Vector<String>();
  + Membuat instance Vector yang menyimpan elemen dengan tipe data String. Vector adalah kelas yang mirip dengan ArrayList, tetapi berbeda dalam hal sinkronisasi (thread-safe) dan beberapa fitur lainnya.

1. Menambahkan Elemen ke Vector

* vc.add("Vector Object 1");
  + Menambahkan elemen "Vector Object 1" ke Vector.
* vc.add("Vector Object 2");
  + Menambahkan elemen "Vector Object 2" ke Vector.
* vc.add("Vector Object 3");
  + Menambahkan elemen "Vector Object 3" ke Vector.
* vc.add("Vector Object 4");
  + Menambahkan elemen "Vector Object 4" ke Vector.
* vc.add("Vector Object 5");
  + Menambahkan elemen "Vector Object 5" ke Vector.
* vc.add(3, "Element at fix position");
  + Menambahkan elemen "Element at fix position" pada indeks ke-3 di Vector. Elemen yang ada pada dan setelah indeks 3 akan digeser ke kanan.

1. Menampilkan Ukuran dan Elemen dari Vector

* System.out.println("Vector Size :" + vc.size());
  + Mencetak ukuran Vector, yaitu jumlah elemen yang ada di dalamnya.
* for(int i = 0; i < vc.size(); i++)
  + Loop untuk iterasi melalui semua elemen dalam Vector.
* System.out.println("Vector Element " + i + " :" + vc.get(i));
  + Mencetak setiap elemen dari Vector beserta indeksnya.
* **Percobaan 6**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.\*;

class IteratorDemo {

public static void main(String args[]) {

// create an array list

ArrayList al = new ArrayList();

// add elements to the array list

al.add("C");

al.add("A");

al.add("E");

al.add("B");

al.add("D");

al.add("F");

// use iterator to display contents of al

System.out.print("Original contents of al: ");

Iterator itr = al.iterator();

while (itr.hasNext()) {

Object element = itr.next();

System.out.print(element + " ");

}

System.out.println();

// modify objects being iterated

ListIterator litr = al.listIterator();

while (litr.hasNext()) {

Object element = litr.next();

litr.set(element + "+");

}

System.out.print("Modified contents of al: ");

itr = al.iterator();

while (itr.hasNext()) {

Object element = itr.next();

System.out.print(element + " ");

}

System.out.println();

// now, display the list backwards

System.out.print("Modified list backwards: ");

while (litr.hasPrevious()) {

Object element = litr.previous();

System.out.print(element + " ");

}

System.out.println();

}

}

**Output :**



1. Deskripsi Kode

Kode ini mendemonstrasikan penggunaan Iterator dan ListIterator untuk mengiterasi dan memodifikasi elemen dalam sebuah ArrayList di Java. Kode ini mencakup:

* Penambahan elemen ke dalam ArrayList.
* Penggunaan Iterator untuk menampilkan elemen-elemen.
* Penggunaan ListIterator untuk memodifikasi elemen-elemen di dalam list.
* Menampilkan elemen dalam urutan terbalik menggunakan ListIterator.

1. Analisis Kode
2. Penggunaan ArrayList

* ArrayList al = new ArrayList();
  + Membuat instance ArrayList bernama al. ArrayList digunakan untuk menyimpan elemen dengan urutan yang bisa diakses melalui indeks.
* al.add("C");
  + Menambahkan elemen "C" ke dalam ArrayList.
* al.add("A");
  + Menambahkan elemen "A" ke dalam ArrayList.
* al.add("E");
  + Menambahkan elemen "E" ke dalam ArrayList.
* al.add("B");
  + Menambahkan elemen "B" ke dalam ArrayList.
* al.add("D");
  + Menambahkan elemen "D" ke dalam ArrayList.
* al.add("F");
  + Menambahkan elemen "F" ke dalam ArrayList.

1. Penggunaan Iterator untuk Menampilkan Elemen

* Iterator itr = al.iterator();
  + Membuat Iterator untuk ArrayList al. Iterator digunakan untuk mengiterasi elemen satu per satu.
* while (itr.hasNext())
  + Loop yang terus berlanjut selama ada elemen berikutnya.
* Object element = itr.next();
  + Mengambil elemen berikutnya dari iterator.
* System.out.print(element + " ");
  + Mencetak elemen yang diambil.

1. Modifikasi Elemen Menggunakan ListIterator

* ListIterator litr = al.listIterator();
  + Membuat ListIterator untuk ArrayList al. ListIterator memungkinkan iterasi maju dan mundur serta modifikasi elemen.
* while (litr.hasNext())
  + Loop yang terus berlanjut selama ada elemen berikutnya.
* Object element = litr.next();
  + Mengambil elemen berikutnya dari iterator.
* litr.set(element + "+");
  + Mengganti elemen yang sedang diiterasi dengan elemen yang telah dimodifikasi (element + "+").

1. Menampilkan Elemen yang Dimodifikasi

* itr = al.iterator();
  + Membuat ulang Iterator untuk menampilkan elemen yang telah dimodifikasi.
* while (itr.hasNext())
  + Loop yang terus berlanjut selama ada elemen berikutnya.
* Object element = itr.next();
  + Mengambil elemen berikutnya dari iterator.
* System.out.print(element + " ");
  + Mencetak elemen yang diambil.

1. Menampilkan List Secara Terbalik

* while (litr.hasPrevious())
  + Loop yang terus berlanjut selama ada elemen sebelumnya. ListIterator memungkinkan iterasi mundur.
* Object element = litr.previous();
  + Mengambil elemen sebelumnya dari iterator.
* System.out.print(element + " ");
  + Mencetak elemen yang diambil.
* **Percobaan 7**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.Vector;

import java.util.Enumeration;

public class EnumerationTester {

public static void main(String args[]) {

Enumeration days;

Vector dayNames = new Vector();

dayNames.add("Sunday");

dayNames.add("Monday");

dayNames.add("Tuesday");

dayNames.add("Wednesday");

dayNames.add("Thursday");

dayNames.add("Friday");

dayNames.add("Saturday");

days = dayNames.elements();

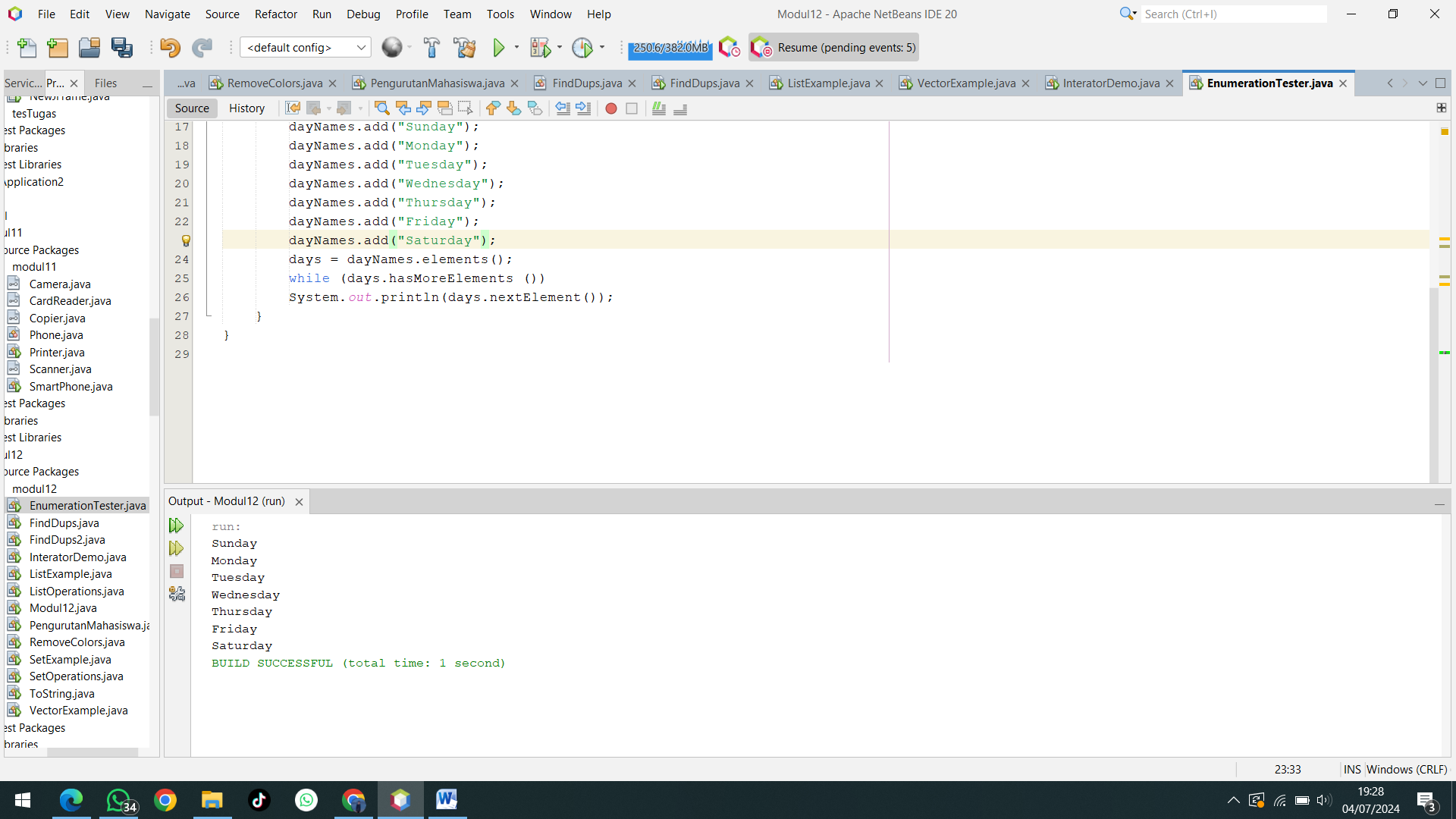
while (days.hasMoreElements ())

System.out.println(days.nextElement());

}

}

**Output :**



**Penjelasan Kode**

1. Import Statements:

* import java.util.Vector;: Mengimpor kelas Vector dari paket java.util.
* import java.util.Enumeration;: Mengimpor antarmuka Enumeration dari paket java.util.

1. Class Declaration:

* public class EnumerationTester: Mendeklarasikan kelas publik bernama EnumerationTester.

1. Main Method:

* public static void main(String args[]): Mendeklarasikan metode utama yang akan dijalankan saat program dijalankan.

1. Declaration of Variables:

* Enumeration days;: Mendeklarasikan variabel days dari tipe Enumeration.
* Vector dayNames = new Vector();: Membuat instance dari Vector bernama dayNames.

1. Adding Elements to the Vector:

* dayNames.add("Sunday");: Menambahkan elemen "Sunday" ke dayNames.
* dayNames.add("Monday");: Menambahkan elemen "Monday" ke dayNames.
* dayNames.add("Tuesday");: Menambahkan elemen "Tuesday" ke dayNames.
* dayNames.add("Wednesday");: Menambahkan elemen "Wednesday" ke dayNames.
* dayNames.add("Thursday");: Menambahkan elemen "Thursday" ke dayNames.
* dayNames.add("Friday");: Menambahkan elemen "Friday" ke dayNames.
* dayNames.add("Saturday");: Menambahkan elemen "Saturday" ke dayNames.

1. Getting Enumeration of Elements:

* days = dayNames.elements();: Mengambil enumerasi elemen-elemen dari dayNames dan menyimpannya dalam variabel days.

1. Iterating Over Enumeration:

* while (days.hasMoreElements()): Memulai loop yang akan berjalan selama days masih memiliki elemen berikutnya.
* System.out.println(days.nextElement());: Mencetak elemen berikutnya dari days ke konsol.

**Analisis Kode**

1. Kelas Vector:

* Vector adalah kelas dari paket java.util yang menyediakan array dinamis yang dapat bertambah secara otomatis ketika elemen ditambahkan.
* Vector adalah thread-safe, yang berarti setiap metode disinkronisasi untuk membuatnya aman digunakan di lingkungan multi-threaded. Namun, ini membuat Vector agak lebih lambat dibandingkan dengan ArrayList yang tidak sinkron.

1. Antarmuka Enumeration:

* Enumeration adalah antarmuka dari paket java.util yang menyediakan metode untuk mengambil elemen satu per satu dari struktur data seperti Vector.
* Metode utama dari Enumeration adalah hasMoreElements() yang mengembalikan true jika masih ada elemen berikutnya dan nextElement() yang mengembalikan elemen berikutnya.

1. Menambahkan Elemen ke Vector:

* dayNames.add("Sunday"); hingga dayNames.add("Saturday"); menambahkan nama-nama hari ke dalam Vector.

1. Mengambil dan Menggunakan Enumeration:

* days = dayNames.elements(); mengambil enumerasi elemen-elemen dari Vector dayNames.
* while (days.hasMoreElements()) dan System.out.println(days.nextElement()); digunakan untuk mencetak setiap elemen dari enumerasi ke konsol.
* **Percobaan 8**

package modul12;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Enumeration;

import java.util.Vector;

/\*\*

\*

\* @author Willy

\*/

public class CreateArrayListFromEnumerationExample {

public static void main(String[] args) {

//create a Vector object

Vector v = new Vector();

//Add elements to Vector

v.add("A");

v.add("B");

v.add("D");

v.add("E");

v.add("F");

System.out.println("Vector contains : " + v);

//Get Enumeration over Vector

Enumeration e = v.elements();

//Create ArrayList from Enumeration of Vector

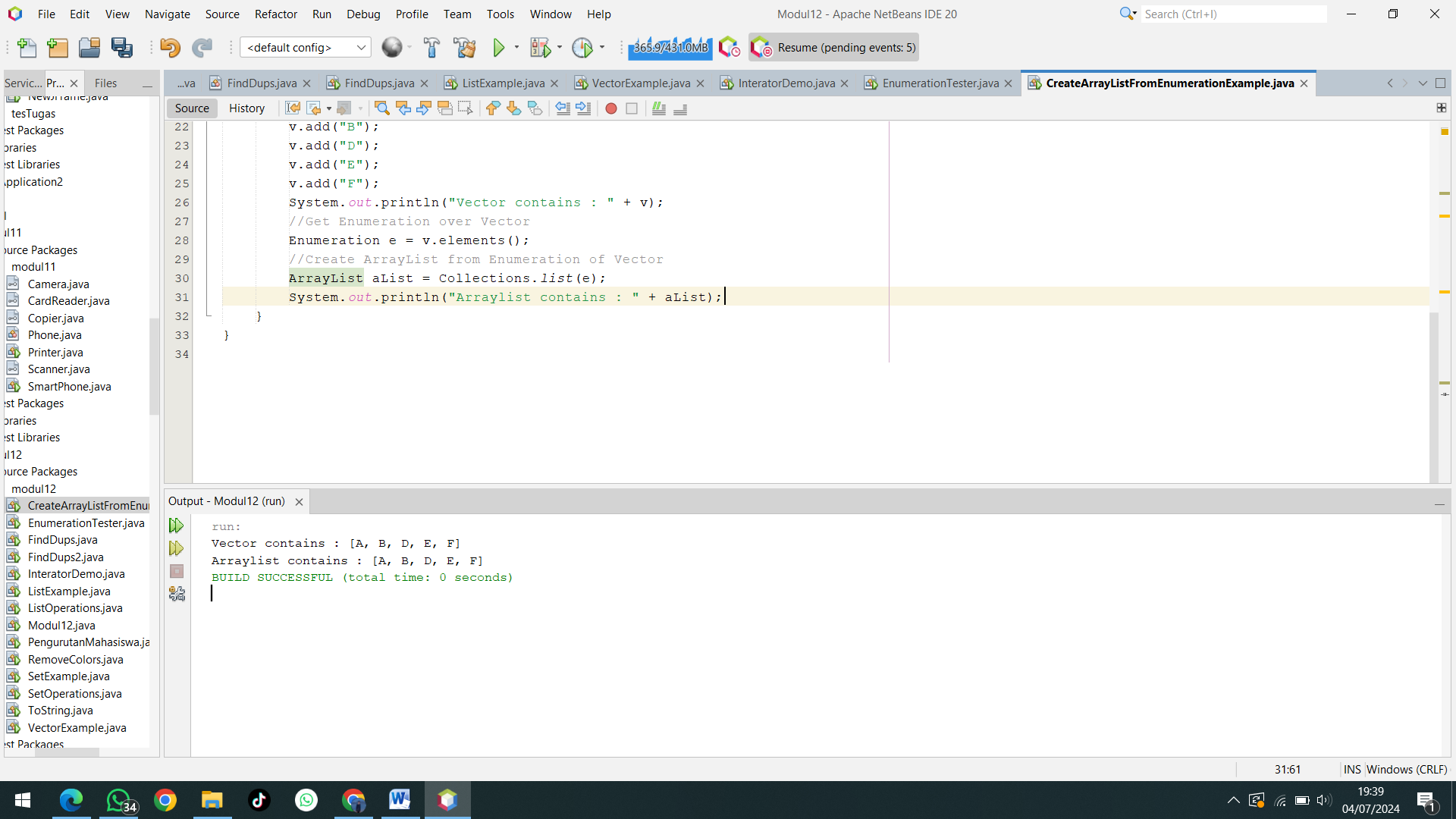
ArrayList aList = Collections.list(e);

System.out.println("Arraylist contains : " + aList);

}

}

**Output :**



**Penjelasan Kode**

1. Import Statements:

* Tidak ada impor eksplisit, namun kode ini menggunakan kelas dari paket java.util secara implisit, yaitu Vector, Enumeration, ArrayList, dan Collections.

1. Class Declaration:

* public class CreateArrayListFromEnumerationExample: Mendeklarasikan kelas publik bernama CreateArrayListFromEnumerationExample.

1. Main Method:

* public static void main(String[] args): Mendeklarasikan metode utama yang akan dijalankan saat program dijalankan.

1. Creating and Initializing Vector:

* Vector v = new Vector();: Membuat instance dari Vector bernama v.
* v.add("A"); hingga v.add("F");: Menambahkan elemen-elemen "A", "B", "D", "E", dan "F" ke dalam Vector v.

1. Printing Vector Content:

* System.out.println("Vector contains : " + v);: Mencetak konten dari Vector v ke konsol.

1. Getting Enumeration over Vector:

* Enumeration e = v.elements();: Mendapatkan enumerasi elemen-elemen dari Vector v dan menyimpannya dalam variabel e.

1. Creating ArrayList from Enumeration:

* ArrayList aList = Collections.list(e);: Menggunakan metode statis Collections.list() untuk membuat ArrayList dari Enumeration e.

1. Printing ArrayList Content:

* System.out.println("Arraylist contains : " + aList);: Mencetak konten dari ArrayList aList ke konsol.

**Analisis Kode**

1. Kelas Vector:

* Vector adalah kelas dari paket java.util yang menyediakan array dinamis yang dapat bertambah secara otomatis ketika elemen ditambahkan.
* Vector adalah thread-safe, yang berarti setiap metode disinkronisasi untuk membuatnya aman digunakan di lingkungan multi-threaded. Namun, ini membuat Vector agak lebih lambat dibandingkan dengan ArrayList yang tidak sinkron.

1. Antarmuka Enumeration:

* Enumeration adalah antarmuka dari paket java.util yang menyediakan metode untuk mengambil elemen satu per satu dari struktur data seperti Vector.
* Metode utama dari Enumeration adalah hasMoreElements() yang mengembalikan true jika masih ada elemen berikutnya dan nextElement() yang mengembalikan elemen berikutnya.

1. Membuat ArrayList dari Enumeration:

* Collections.list(Enumeration e) adalah metode statis yang mengonversi Enumeration ke dalam ArrayList. Ini memungkinkan kita untuk menggunakan metode modern dari ArrayList dengan elemen yang awalnya ada dalam Enumeration.
* **Percobaan 9**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.ArrayList;

import java.util.Collections;

import java.util.Vector;

public class CopyElementsOfArrayListToVectorExample {

public static void main(String[] args) {

//create an ArrayList object

ArrayList<String> arrayList = new ArrayList<>();

//Add elements to ArrayList

arrayList.add("1");

arrayList.add("4");

arrayList.add("2");

arrayList.add("5");

arrayList.add("3");

//create a Vector object

Vector<String> v = new Vector<>();

//Add elements to Vector

v.add("A");

v.add("B");

v.add("D");

v.add("E");

v.add("F");

v.add("G");

v.add("H");

System.out.println("Before copy, Vector Contains : " + v);

// Ensure the Vector is large enough to copy the elements of ArrayList

for (int i = 0; i < arrayList.size(); i++) {

v.add(""); // Add empty elements to Vector

}

// Copy all elements of ArrayList to Vector using copy method of Collections class

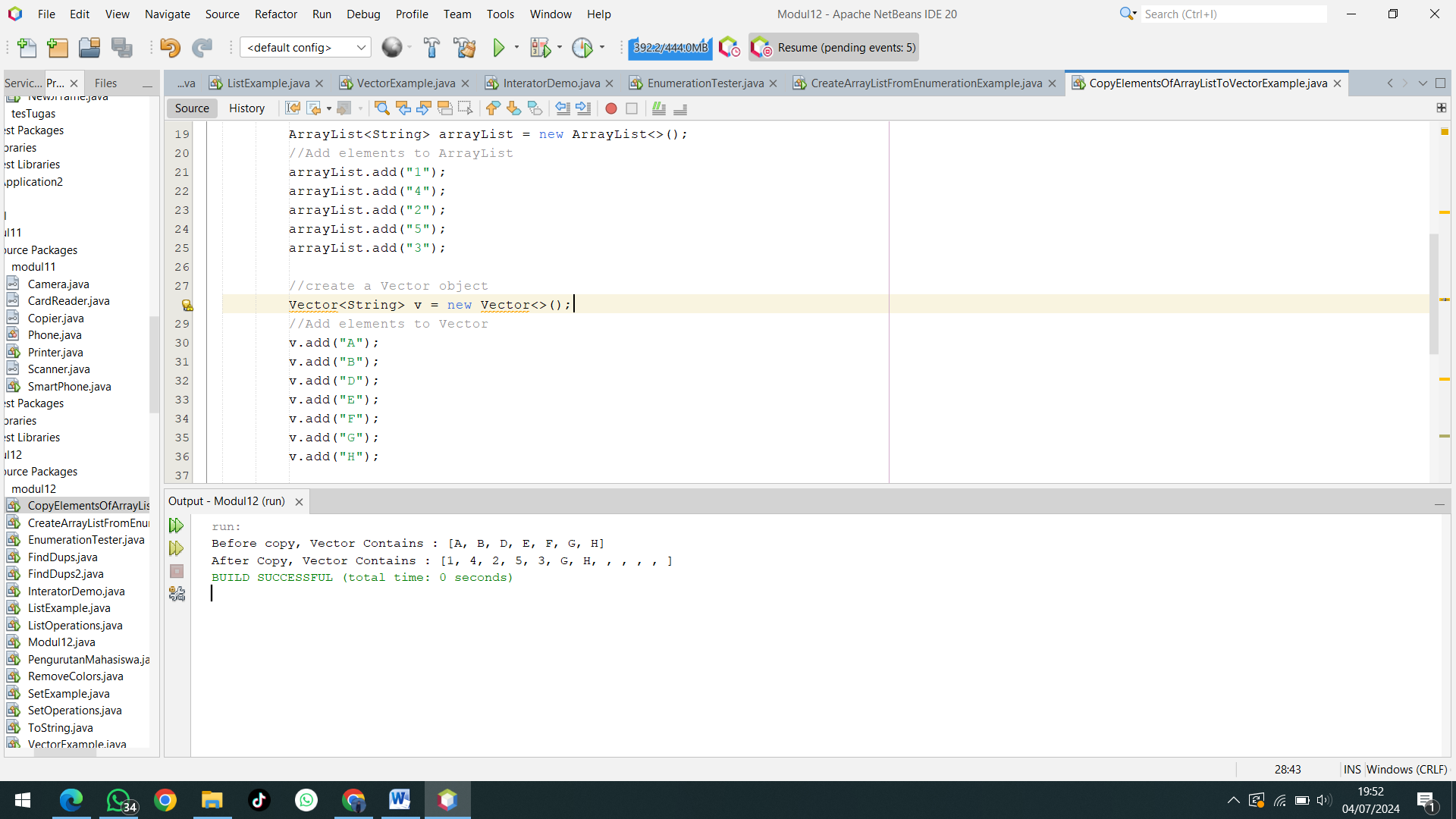
Collections.copy(v, arrayList);

System.out.println("After Copy, Vector Contains : " + v);

}

}

**Output :**



**Penjelasan Kode**

1. Import Statements:

* import java.util.ArrayList;
* import java.util.Collections;
* import java.util.Vector;

Mengimpor kelas-kelas yang diperlukan dari paket java.util.

1. Class Declaration:

* public class CopyElementsOfArrayListToVectorExample: Mendeklarasikan kelas publik bernama CopyElementsOfArrayListToVectorExample.

1. Main Method:

* public static void main(String[] args): Mendeklarasikan metode utama yang akan dijalankan saat program dijalankan.

1. Creating and Initializing ArrayList:

* ArrayList<String> arrayList = new ArrayList<>();: Membuat instance dari ArrayList bernama arrayList dengan tipe String.
* arrayList.add("1"); hingga arrayList.add("3");: Menambahkan elemen-elemen "1", "4", "2", "5", dan "3" ke dalam ArrayList arrayList.

1. Creating and Initializing Vector:

* Vector<String> v = new Vector<>();: Membuat instance dari Vector bernama v dengan tipe String.
* v.add("A"); hingga v.add("H");: Menambahkan elemen-elemen "A", "B", "D", "E", "F", "G", dan "H" ke dalam Vector v.

1. Printing Vector Content Before Copy:

* System.out.println("Before copy, Vector Contains : " + v);: Mencetak konten dari Vector v ke konsol sebelum operasi salin dilakukan.

1. Ensuring Vector Size:

* for (int i = 0; i < arrayList.size(); i++) { v.add(""); }: Menambahkan elemen kosong ("") ke dalam Vector untuk memastikan ukurannya cukup besar untuk menampung semua elemen dari ArrayList. Hal ini dilakukan karena Collections.copy() membutuhkan Vector tujuan memiliki ukuran yang setidaknya sama dengan ukuran ArrayList sumber.

1. Copying Elements from ArrayList to Vector:

* Collections.copy(v, arrayList);: Menyalin semua elemen dari ArrayList arrayList ke Vector v menggunakan metode statis Collections.copy().

1. Printing Vector Content After Copy:

* System.out.println("After Copy, Vector Contains : " + v);: Mencetak konten dari Vector v ke konsol setelah operasi salin dilakukan.

**Analisis Kode**

1. Penggunaan Generics:

* Menggunakan ArrayList<String> dan Vector<String> untuk menghindari peringatan unchecked dan membuat kode lebih aman.

1. Menangani Ukuran Vector:

* Menambahkan elemen kosong ke Vector untuk memastikan ukurannya cukup besar sebelum menggunakan Collections.copy(). Hal ini penting karena Collections.copy() akan menimpa elemen-elemen di Vector tujuan, dan Vector harus memiliki ukuran yang cukup besar untuk menampung semua elemen dari ArrayList.

1. Menggunakan Collections.copy():

* Collections.copy(v, arrayList); menyalin elemen-elemen dari ArrayList ke Vector. Metode ini menimpa elemen-elemen di Vector tujuan dengan elemen-elemen dari ArrayList sumber.
* **Percobaan 10**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.ArrayList;

import java.util.Vector;

public class AppendAllElementsOfOtherCollectionToArrayListExample {

public static void main(String[] args) {

//create an ArrayList object

ArrayList<String> arrayList = new ArrayList<>();

//Add elements to ArrayList

arrayList.add("1");

arrayList.add("2");

arrayList.add("3");

//create a new Vector object

Vector<String> v = new Vector<>();

v.add("4");

v.add("5");

//append all elements of Vector to ArrayList

arrayList.addAll(v);

//display elements of ArrayList

System.out.println("After appending all elements of Vector, ArrayList contains:");

for (String element : arrayList) {

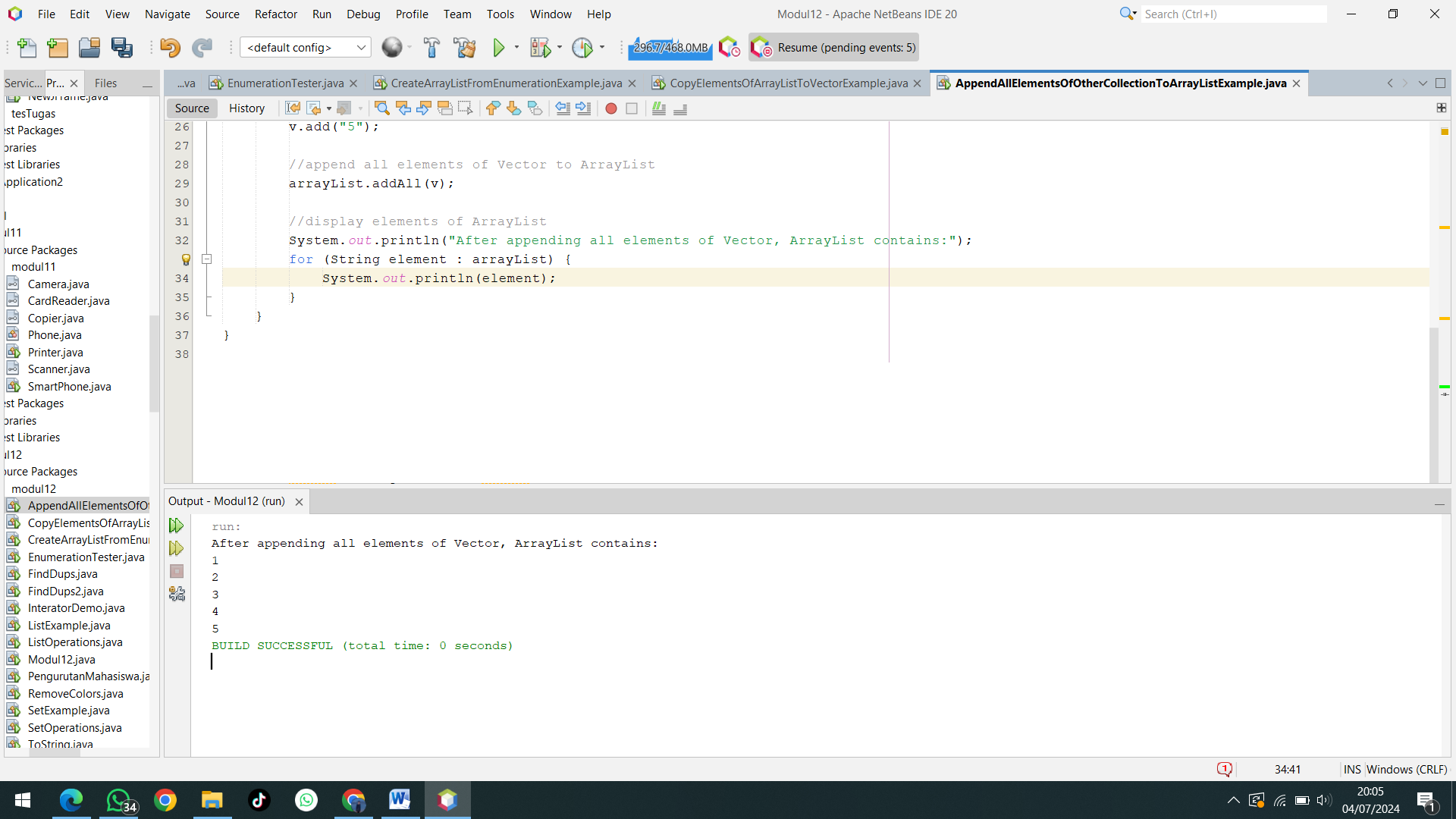
System.out.println(element);

}

}

}

**Output :**



**Penjelasan Kode**

1. Import Statements:

* import java.util.ArrayList;
* import java.util.Vector;

Mengimpor kelas ArrayList dan Vector dari paket java.util yang diperlukan untuk implementasi koleksi.

1. Class Declaration:

* public class AppendAllElementsOfOtherCollectionToArrayListExample: Mendeklarasikan kelas publik bernama AppendAllElementsOfOtherCollectionToArrayListExample.

1. Main Method:

* public static void main(String[] args): Mendeklarasikan metode utama yang akan dieksekusi saat program dijalankan.

1. Creating and Initializing ArrayList:

* ArrayList<String> arrayList = new ArrayList<>();: Membuat instance ArrayList dengan tipe String.
* arrayList.add("1"); hingga arrayList.add("3");: Menambahkan elemen-elemen "1", "2", dan "3" ke dalam ArrayList.

1. Creating and Initializing Vector:

* Vector<String> v = new Vector<>();: Membuat instance Vector dengan tipe String.

v.add("4"); dan v.add("5");: Menambahkan elemen-elemen "4" dan "5" ke dalam Vector.

1. Appending Elements of Vector to ArrayList:

* arrayList.addAll(v);: Menambahkan semua elemen dari Vector v ke dalam ArrayList arrayList. Metode addAll dari ArrayList memungkinkan kita untuk menambahkan semua elemen dari koleksi lain ke ArrayList.

1. Displaying Elements of ArrayList:

* System.out.println("After appending all elements of Vector, ArrayList contains:");: Mencetak pesan untuk menunjukkan bahwa elemen-elemen dari Vector telah ditambahkan ke ArrayList.
* for (String element : arrayList) { System.out.println(element); }: Menggunakan enhanced for loop untuk iterasi melalui semua elemen dalam ArrayList dan mencetak setiap elemen.

**Analisis Kode**

1. Penggunaan Generics:

* Menggunakan generics untuk mendeklarasikan ArrayList dan Vector dengan tipe String untuk memastikan keamanan tipe dan menghindari peringatan unchecked.

1. Penambahan Elemen ke ArrayList:

* Elemen-elemen "1", "2", dan "3" ditambahkan ke ArrayList menggunakan metode add.
* Elemen-elemen "4" dan "5" ditambahkan ke Vector menggunakan metode add.

1. Penggabungan Elemen:

* Metode addAll digunakan untuk menambahkan semua elemen dari Vector ke ArrayList. Ini adalah cara yang efisien untuk menggabungkan dua koleksi.
* **Percobaan 11**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.PriorityQueue;

public class PriorityQueueDemo {

public static void main(String[] args) {

PriorityQueue<String> stringQueue = new PriorityQueue<>();

stringQueue.add("ab");

stringQueue.add("abcd");

stringQueue.add("abc");

stringQueue.add("a");

// don't use iterator which may or may not

// show the PriorityQueue's order

while (stringQueue.size() > 0) {

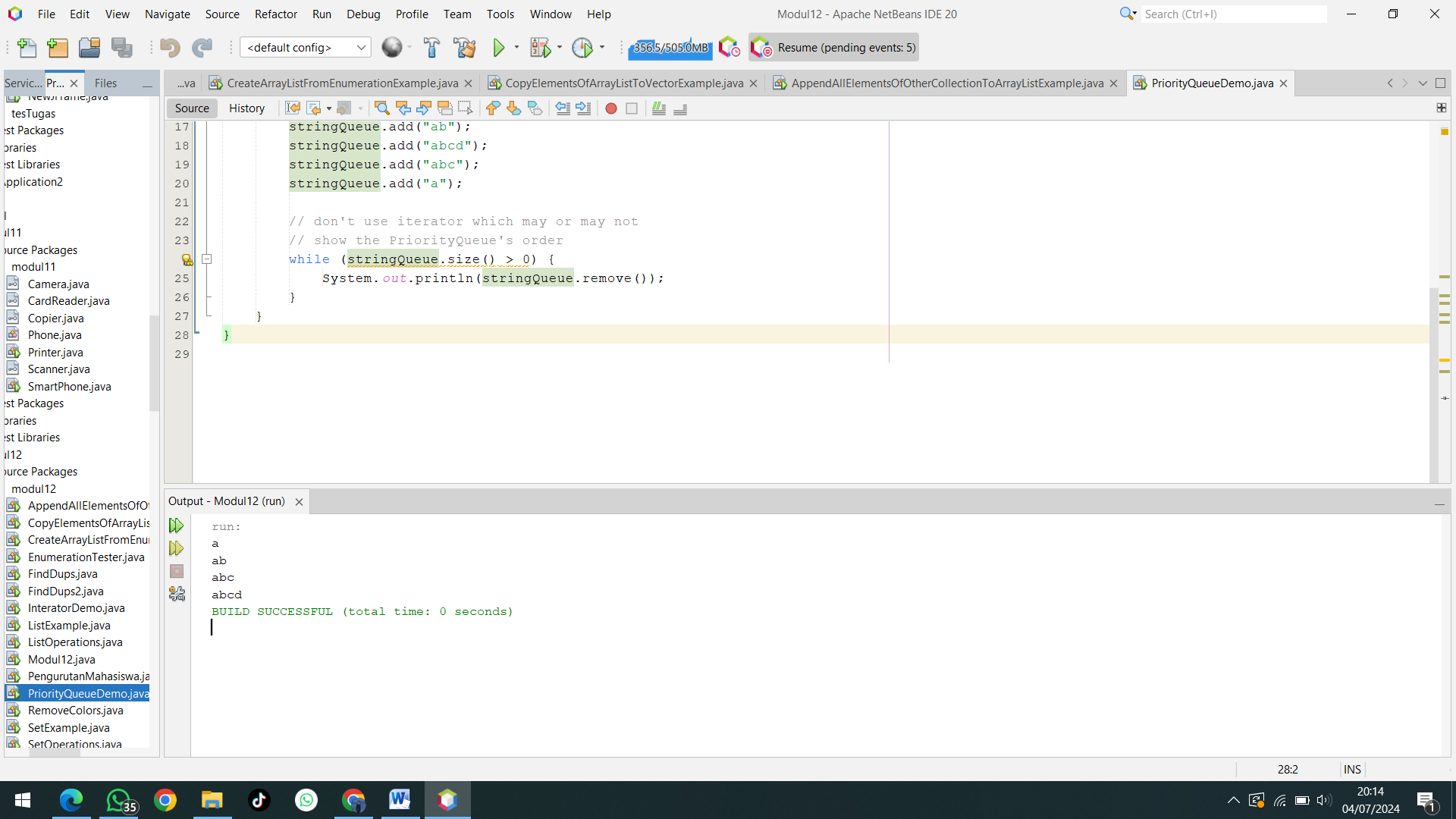
System.out.println(stringQueue.remove());

}

}

}

**Output :**



**Penjelasan Kode**

1. Import Statements:

* import java.util.PriorityQueue;

Mengimpor kelas PriorityQueue dari paket java.util. PriorityQueue adalah implementasi dari struktur data antrian prioritas, yang secara otomatis mengurutkan elemen berdasarkan urutan prioritas alami atau comparator yang diberikan.

1. Class Declaration:

* public class PriorityQueueDemo

Mendeklarasikan kelas publik bernama PriorityQueueDemo.

1. Main Method:

* public static void main(String[] args)

Mendeklarasikan metode utama yang akan dieksekusi saat program dijalankan.

1. Creating and Initializing PriorityQueue:

* PriorityQueue<String> stringQueue = new PriorityQueue<>();

Membuat instance PriorityQueue dengan tipe String. Antrian prioritas ini akan mengurutkan elemen secara leksikografis karena elemen-elemennya adalah String.

1. Adding Elements to PriorityQueue:

* stringQueue.add("ab");
* stringQueue.add("abcd");
* stringQueue.add("abc");
* stringQueue.add("a");

Menambahkan elemen-elemen "ab", "abcd", "abc", dan "a" ke dalam PriorityQueue. Elemen-elemen ini akan diurutkan secara otomatis berdasarkan urutan prioritas alami String.

1. Displaying and Removing Elements from PriorityQueue:

* while (stringQueue.size() > 0)

Menggunakan loop while untuk mengeluarkan dan menampilkan elemen dari PriorityQueue hingga kosong.

* System.out.println(stringQueue.remove());

Mengeluarkan dan menampilkan elemen dari PriorityQueue. Metode remove() mengeluarkan elemen dengan prioritas tertinggi (dalam kasus ini, elemen dengan urutan leksikografis terendah) dari antrian.

* **Percobaan 12**

package modul12;

/\*\*

\*

\* @author Willy

\*/

import java.util.Comparator;

import java.util.PriorityQueue;

public class PQueueTest {

public static void main(String[] args) {

PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>(10, new Comparator<Integer>() {

@Override

public int compare(Integer int1, Integer int2) {

boolean flag1 = isPrime(int1);

boolean flag2 = isPrime(int2);

if (flag1 == flag2) {

return int1.compareTo(int2);

} else if (flag1) {

return -1;

} else {

return 1;

}

}

});

pQueue.add(1);

pQueue.add(5);

pQueue.add(6);

pQueue.add(4);

pQueue.add(2);

pQueue.add(9);

pQueue.add(7);

pQueue.add(8);

pQueue.add(10);

pQueue.add(3);

while (!pQueue.isEmpty()) {

Integer head = pQueue.poll();

System.out.print(head + " <-- ");

}

}

public static boolean isPrime(int n) {

if (n <= 1) {

return false;

}

if (n == 2) {

return true;

}

if (n % 2 == 0) {

return false;

}

long m = (long) Math.sqrt(n);

for (long i = 3; i <= m; i += 2) {

if (n % i == 0) {

return false;

}

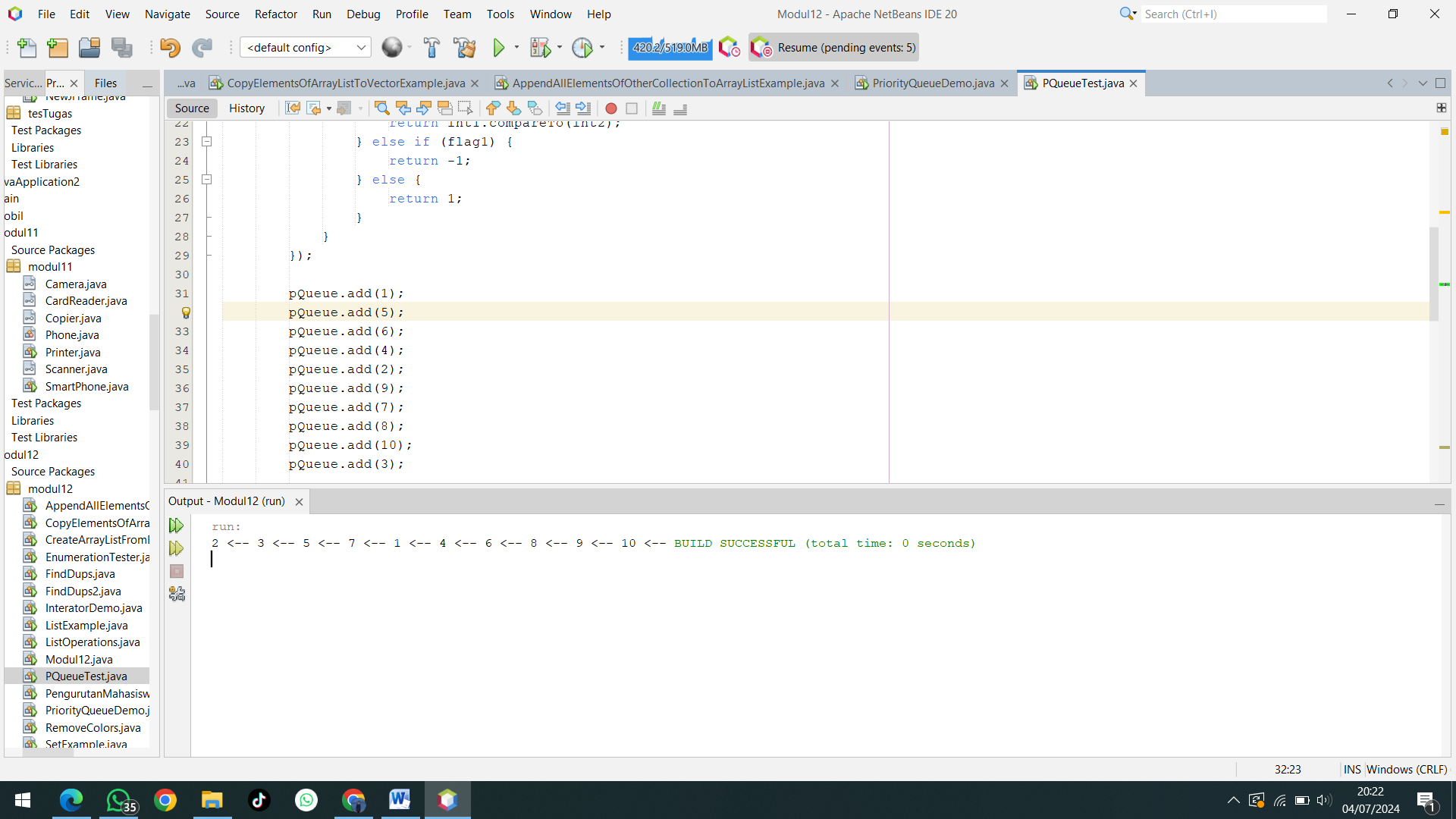
}

return true;

}

}

**Output :**



Tujuan

Program ini bertujuan untuk menunjukkan penggunaan PriorityQueue dengan comparator kustom yang memprioritaskan angka berdasarkan apakah mereka adalah bilangan prima, dan kemudian berdasarkan urutan alami jika status primanya sama.

1. Impor:

**import java.util.Comparator;**

**import java.util.PriorityQueue;**

Mengimpor kelas-kelas yang diperlukan untuk menggunakan PriorityQueue dan membuat Comparator.

1. Metode ‘main’:

**public static void main(String[] args) {**

**PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>(10, new Comparator<Integer>() {**

**@Override**

**public int compare(Integer int1, Integer int2) {**

**boolean flag1 = isPrime(int1);**

**boolean flag2 = isPrime(int2);**

**if (flag1 == flag2) {**

**return int1.compareTo(int2);**

**} else if (flag1) {**

**return -1;**

**} else {**

**return 1;**

**}**

**}**

**});**

* Inisialisasi PriorityQueue:
  + Menginisialisasi PriorityQueue dengan kapasitas awal 10.
  + Menggunakan Comparator kustom untuk menentukan prioritas elemen.
* Logika Comparator:
  + Pemeriksaan Bilangan Prima:
  + isPrime(int1) dan isPrime(int2) digunakan untuk memeriksa apakah angka tersebut bilangan prima.
* Aturan Perbandingan:
  + Jika kedua angka adalah bilangan prima atau bukan bilangan prima, mereka dibandingkan secara alami (int1.compareTo(int2)).
  + Jika satu angka bilangan prima dan yang lainnya bukan, angka bilangan prima diprioritaskan.

**pQueue.add(1);**

**pQueue.add(5);**

**pQueue.add(6);**

**pQueue.add(4);**

**pQueue.add(2);**

**pQueue.add(9);**

**pQueue.add(7);**

**pQueue.add(8);**

**pQueue.add(10);**

**pQueue.add(3);**

* Menambahkan angka-angka ke dalam antrian prioritas.

**while (!pQueue.isEmpty()) {**

**Integer head = pQueue.poll();**

**System.out.print(head + " <-- ");**

**}**

Mengambil elemen dengan prioritas tertinggi (bilangan prima terlebih dahulu, diikuti oleh bilangan bukan prima) dan mencetaknya hingga antrian kosong.

1. Metode ‘isPrime’:

**public static boolean isPrime(int n) {**

**if (n <= 1) {**

**return false;**

**}**

**if (n == 2) {**

**return true;**

**}**

**if (n % 2 == 0) {**

**return false;**

**}**

**long m = (long) Math.sqrt(n);**

**for (long i = 3; i <= m; i += 2) {**

**if (n % i == 0) {**

**return false;**

**}**

**}**

**return true;**

**}**

* Fungsi Pemeriksaan Bilangan Prima:
  + Memeriksa apakah sebuah angka adalah bilangan prima.
  + Efisien dengan memeriksa hanya hingga akar kuadrat dari angka dan menggunakan langkah 2 pada loop.

1. **Latian**

* **Latihan 1**

package modul12;

/\*\*

\*

\* @author Willy

\*/

//LATIHAN 1

import java.util.HashSet;

import java.util.Set;

public class SetOperations {

public static void main(String[] args) {

// Definisikan himpunan A dan B

Set<Integer> A = new HashSet<>();

Set<Integer> B = new HashSet<>();

// Isi himpunan A

A.add(1);

A.add(2);

A.add(3);

A.add(4);

A.add(5);

// Isi himpunan B

B.add(5);

B.add(6);

B.add(7);

B.add(8);

B.add(9);

B.add(10);

// Operasi - A - B (Perbedaan)

Set<Integer> difference = new HashSet<>(A);

difference.removeAll(B);

System.out.println("A - B: " + difference);

// Operasi - A ∩ B (Irisan)

Set<Integer> intersection = new HashSet<>(A);

intersection.retainAll(B);

System.out.println("A ∩ B: " + intersection);

// Operasi - A U B (Gabungan)

Set<Integer> union = new HashSet<>(A);

union.addAll(B);

System.out.println("A U B: " + union);

// Operasi - A ⊆ B (Subset)

boolean isSubset = B.containsAll(A);

System.out.println("A ⊆ B: " + isSubset);

}

}

**Output :**



**Analisis Kode**

**Tujuan** Program ini bertujuan untuk menunjukkan operasi dasar pada himpunan (set) menggunakan kelas HashSet di Java. Operasi yang dilakukan meliputi perbedaan, irisan, gabungan, dan pengecekan subset antara dua himpunan.

**Komponen Utama**

1. Definisi Himpunan:

**Set<Integer> A = new HashSet<>();**

**Set<Integer> B = new HashSet<>();**

* Set: Interface yang digunakan untuk menyimpan elemen yang unik, tanpa urutan tertentu.
* HashSet: Implementasi dari Set yang menyimpan elemen tanpa urutan dan tidak memungkinkan duplikasi.

1. Mengisi Himpunan:

**// Isi himpunan A**

**A.add(1);**

**A.add(2);**

**A.add(3);**

**A.add(4);**

**A.add(5);**

**// Isi himpunan B**

**B.add(5);**

**B.add(6);**

**B.add(7);**

**B.add(8);**

**B.add(9);**

**B.add(10);**

* Himpunan A berisi angka dari 1 hingga 5.
* Himpunan B berisi angka dari 5 hingga 10.

1. Operasi Himpunan:

* Perbedaan (A-B):

**Set<Integer> difference = new HashSet<>(A);**

**difference.removeAll(B);**

**System.out.println("A - B: " + difference);**

* + difference: Menghitung elemen yang ada di A tetapi tidak ada di B.
  + removeAll(B): Menghapus semua elemen di B dari himpunan difference.
  + Output: A - B: [1, 2, 3, 4]
* Irisan (A ∩ B):

**Set<Integer> intersection = new HashSet<>(A);**

**intersection.retainAll(B);**

**System.out.println("A ∩ B: " + intersection);**

* intersection: Menghitung elemen yang ada di kedua himpunan A dan B.
* retainAll(B): Mempertahankan hanya elemen yang juga ada di B.
* Output: A ∩ B: [5]
* Gabungan (A U B):

**Set<Integer> union = new HashSet<>(A);**

**union.addAll(B);**

**System.out.println("A U B: " + union);**

* union: Menghitung semua elemen yang ada di A atau di B atau di kedua-duanya.
* addAll(B): Menambahkan semua elemen dari B ke himpunan union.
* Output: A U B: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
* Subset (A ⊆ B):

**boolean isSubset = B.containsAll(A);**

**System.out.println("A ⊆ B: " + isSubset);**

* isSubset: Mengecek apakah semua elemen di A juga ada di B.
* containsAll(A): Mengembalikan true jika B mengandung semua elemen di A.
* Output: A ⊆ B: false
* **Latihan 2**

package modul12;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

\*

\* @author Willy

\*/

public class ListExample {

public static void main(String[] args) {

// Membuat objek List dengan data bertipe String

List<String> dataList = new ArrayList<>();

dataList.add("Apple");

dataList.add("Banana");

dataList.add("Cherry");

dataList.add("Date");

dataList.add("Elderberry");

// 1. Tampilkan data yang terdapat pada list

System.out.println("Data awal dalam List:");

System.out.println(dataList);

// 2. Baliklah data yang terdapat pada list dan tampilkan

Collections.reverse(dataList);

System.out.println("Data setelah dibalik:");

System.out.println(dataList);

// 3. Acaklah data tersebut dan tampilkan

Collections.shuffle(dataList);

System.out.println("Data setelah diacak:");

System.out.println(dataList);

// 4. Urutkan data tersebut dan tampilkan

Collections.sort(dataList);

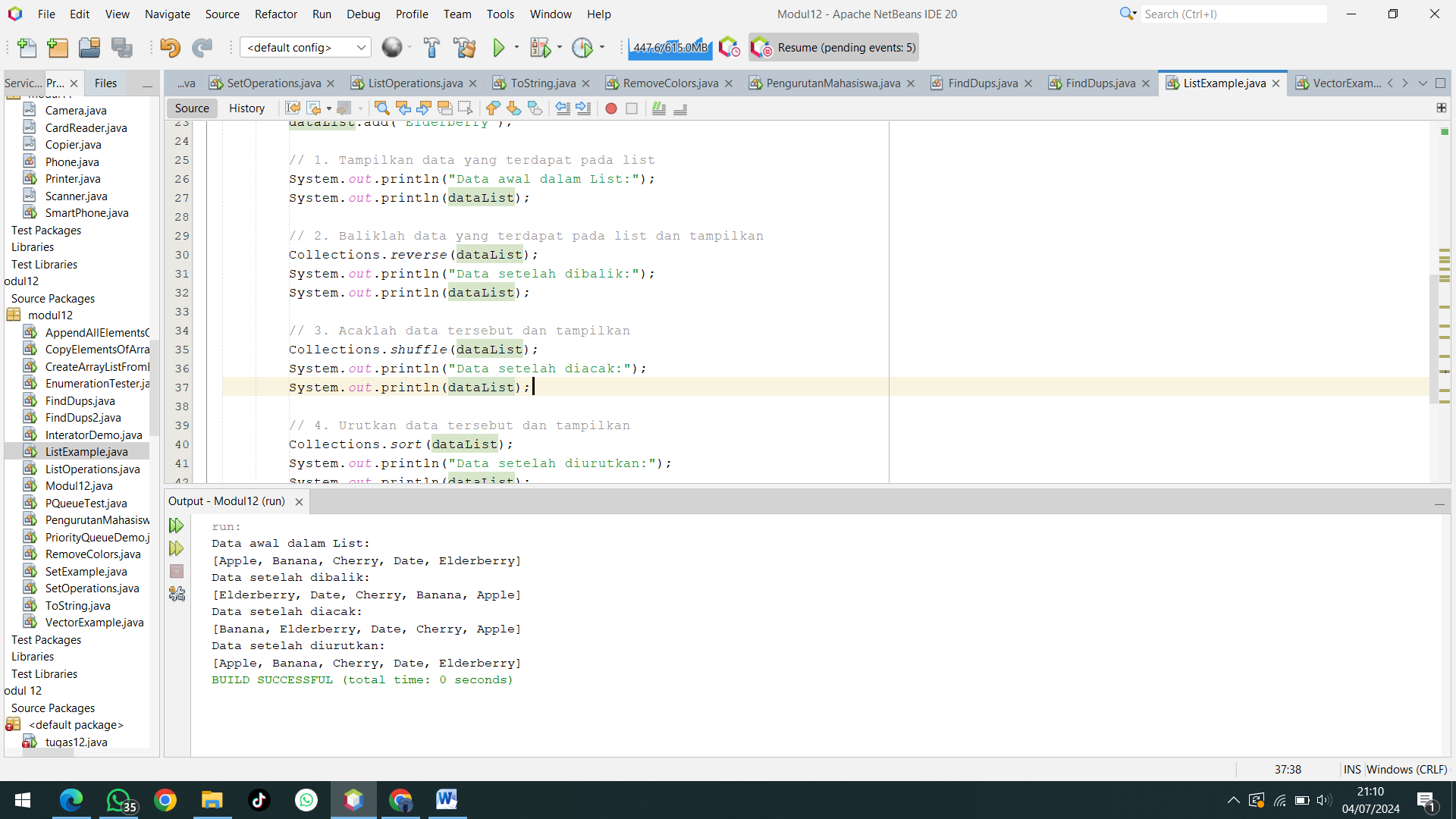
System.out.println("Data setelah diurutkan:");

System.out.println(dataList);

}

}

**Output :**



1. Deklarasi dan Inisialisasi List

* List<String> dataList = new ArrayList<>();: Mendeklarasikan dan menginisialisasi objek ArrayList yang menyimpan elemen-elemen bertipe String. ArrayList adalah implementasi dari interface List yang mendukung operasi dinamis pada list.

1. Menambahkan Data

* dataList.add(...): Menambahkan elemen-elemen string ke dalam list. Elemen yang ditambahkan adalah "Apple", "Banana", "Cherry", "Date", dan "Elderberry".

1. Menampilkan Data Awal

* System.out.println("Data awal dalam List:");: Menampilkan header untuk data awal.
* System.out.println(dataList);: Menampilkan konten dari dataList sebelum operasi apa pun diterapkan. Outputnya akan sesuai dengan urutan elemen yang ditambahkan:

**Data awal dalam List:**

**[Apple, Banana, Cherry, Date, Elderberry]**

1. Membalik Data

* Collections.reverse(dataList);: Membalik urutan elemen-elemen dalam dataList. Setelah dibalik, elemen-elemen akan muncul dalam urutan terbalik dari urutan awal.
* System.out.println("Data setelah dibalik:");: Menampilkan header untuk data yang telah dibalik.
* System.out.println(dataList);: Menampilkan konten dataList setelah dibalik. Outputnya akan:

**Data setelah dibalik:**

**[Elderberry, Date, Cherry, Banana, Apple]**

1. Mengacak Data

* Collections.shuffle(dataList);: Mengacak urutan elemen-elemen dalam dataList. Output dari operasi ini akan bervariasi setiap kali program dijalankan.
* System.out.println("Data setelah diacak:");: Menampilkan header untuk data yang telah diacak.
* System.out.println(dataList);: Menampilkan konten dataList setelah diacak. Outputnya mungkin seperti:

**Data setelah diacak:**

**[Cherry, Banana, Date, Elderberry, Apple] // Contoh output, urutan bisa berbeda**

1. Mengurutkan Data

* Collections.sort(dataList);: Mengurutkan elemen-elemen dalam dataList secara alami, yaitu berdasarkan urutan alfabet untuk string.
* System.out.println("Data setelah diurutkan:");: Menampilkan header untuk data yang telah diurutkan.
* System.out.println(dataList);: Menampilkan konten dataList setelah diurutkan. Outputnya akan:

**Data setelah diurutkan:**

**[Apple, Banana, Cherry, Date, Elderberry]**

* **Latihan 3**

package modul12;

/\*\*

\*

\* @author Willy

\*/

//LATIHAN 3

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

class Mahasiswa implements Comparable<Mahasiswa> {

private String nrp;

private String nama;

public Mahasiswa(String nrp, String nama) {

this.nrp = nrp;

this.nama = nama;

}

public String getNrp() {

return nrp;

}

public String getNama() {

return nama;

}

@Override

public int compareTo(Mahasiswa other) {

return this.nrp.compareTo(other.nrp);

}

@Override

public String toString() {

return "Mahasiswa{nrp='" + nrp + "', nama='" + nama + "'}";

}

public static void main(String[] args) {

// Membuat objek List dengan data bertipe Mahasiswa

List<Mahasiswa> mahasiswaList = new ArrayList<>();

mahasiswaList.add(new Mahasiswa("003", "Alice"));

mahasiswaList.add(new Mahasiswa("001", "Bob"));

mahasiswaList.add(new Mahasiswa("002", "Charlie"));

mahasiswaList.add(new Mahasiswa("005", "Dave"));

mahasiswaList.add(new Mahasiswa("004", "Eve"));

// Tampilkan data yang terdapat pada list

System.out.println("Data List: " + mahasiswaList);

// Baliklah data yang terdapat pada list dan tampilkan

List<Mahasiswa> reversedList = new ArrayList<>(mahasiswaList);

Collections.reverse(reversedList);

System.out.println("Reversed List: " + reversedList);

// Acaklah data tersebut dan tampilkan

List<Mahasiswa> shuffledList = new ArrayList<>(mahasiswaList);

Collections.shuffle(shuffledList);

System.out.println("Shuffled List: " + shuffledList);

// Urutkan data tersebut dan tampilkan

List<Mahasiswa> sortedList = new ArrayList<>(mahasiswaList);

Collections.sort(sortedList);

System.out.println("Sorted List: " + sortedList);

// Urutkan data berdasarkan nama menggunakan Comparator

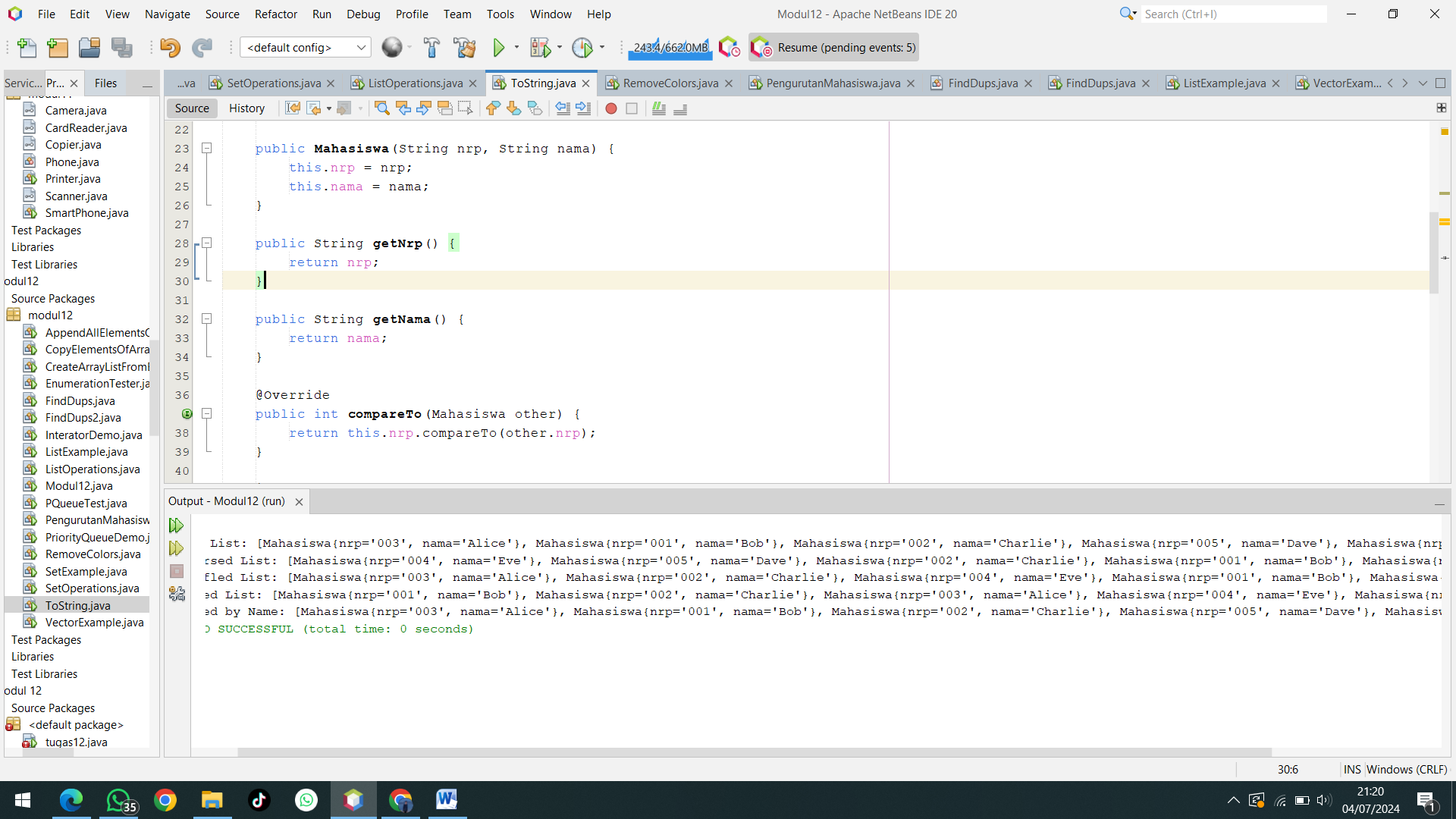
sortedList.sort(Comparator.comparing(Mahasiswa::getNama));

System.out.println("Sorted by Name: " + sortedList);

}

}

**Output :**



1. Deklarasi Kelas Mahasiswa

* Atribut: nrp dan nama adalah atribut dari kelas Mahasiswa, keduanya bertipe String.
* Konstruktor: Mahasiswa(String nrp, String nama) digunakan untuk menginisialisasi objek Mahasiswa.
* Metode Akses: getNrp() dan getNama() mengembalikan nilai atribut nrp dan nama.
* Metode compareTo: Mengimplementasikan Comparable<Mahasiswa> untuk memungkinkan perbandingan objek Mahasiswa berdasarkan nrp. Ini memudahkan pengurutan menggunakan metode Collections.sort() yang akan mengurutkan berdasarkan nrp.
* Metode toString: Mengoverride metode toString untuk menyediakan representasi string yang jelas dari objek Mahasiswa.

1. Metode main

* Inisialisasi List:
  + List<Mahasiswa> mahasiswaList = new ArrayList<>();: Membuat objek ArrayList yang menyimpan elemen bertipe Mahasiswa.
  + Menambahkan beberapa objek Mahasiswa dengan nrp dan nama yang berbeda.
* Menampilkan Data List:
  + System.out.println("Data List: " + mahasiswaList);: Menampilkan list awal yang berisi objek Mahasiswa. Outputnya mengikuti urutan penambahan:

**Data List: [Mahasiswa{nrp='003', nama='Alice'}, Mahasiswa{nrp='001', nama='Bob'}, Mahasiswa{nrp='002', nama='Charlie'}, Mahasiswa{nrp='005', nama='Dave'}, Mahasiswa{nrp='004', nama='Eve'}]**

* Membalik Data:
  + Collections.reverse(reversedList);: Membalik urutan elemen dalam reversedList.
  + System.out.println("Reversed List: " + reversedList);: Menampilkan list yang telah dibalik:

**Reversed List: [Mahasiswa{nrp='004', nama='Eve'}, Mahasiswa{nrp='005', nama='Dave'}, Mahasiswa{nrp='002', nama='Charlie'}, Mahasiswa{nrp='001', nama='Bob'}, Mahasiswa{nrp='003', nama='Alice'}]**

* Mengacak Data:
  + Collections.shuffle(shuffledList);: Mengacak urutan elemen dalam shuffledList.
  + System.out.println("Shuffled List: " + shuffledList);: Menampilkan list yang telah diacak. Output akan berbeda setiap kali program dijalankan:

**Shuffled List: [Mahasiswa{nrp='001', nama='Bob'}, Mahasiswa{nrp='005', nama='Dave'}, Mahasiswa{nrp='004', nama='Eve'}, Mahasiswa{nrp='002', nama='Charlie'}, Mahasiswa{nrp='003', nama='Alice'}]**

* Mengurutkan Data Berdasarkan nrp:
  + Collections.sort(sortedList);: Mengurutkan sortedList berdasarkan nrp karena Mahasiswa mengimplementasikan Comparable.
  + System.out.println("Sorted List: " + sortedList);: Menampilkan list yang telah diurutkan berdasarkan nrp:

**Sorted List: [Mahasiswa{nrp='001', nama='Bob'}, Mahasiswa{nrp='002', nama='Charlie'}, Mahasiswa{nrp='003', nama='Alice'}, Mahasiswa{nrp='004', nama='Eve'}, Mahasiswa{nrp='005', nama='Dave'}]**

* Mengurutkan Data Berdasarkan nama Menggunakan Comparator:
  + sortedList.sort(Comparator.comparing(Mahasiswa::getNama));: Mengurutkan sortedList berdasarkan nama menggunakan Comparator yang membandingkan nama.
  + System.out.println("Sorted by Name: " + sortedList);: Menampilkan list yang diurutkan berdasarkan nama:

**Sorted by Name: [Mahasiswa{nrp='003', nama='Alice'}, Mahasiswa{nrp='001', nama='Bob'}, Mahasiswa{nrp='002', nama='Charlie'}, Mahasiswa{nrp='005', nama='Dave'}, Mahasiswa{nrp='004', nama='Eve'}]**

1. **Tugas**

* **Tugas 1**

package modul12;

/\*\*

\*

\* @author Willy

\*/

//TUGAS 1

import java.util.ArrayList;

import java.util.List;

public class RemoveColors {

public static void main(String[] args) {

// Membuat objek List untuk warna

List<String> warna = new ArrayList<>();

warna.add("MAGENTA");

warna.add("RED");

warna.add("WHITE");

warna.add("BLUE");

warna.add("CYAN");

// Membuat objek List untuk warna yang dihapus

List<String> warnaDihapus = new ArrayList<>();

warnaDihapus.add("RED");

warnaDihapus.add("WHITE");

warnaDihapus.add("BLUE");

// Menampilkan warna sebelum penghapusan

System.out.println("Warna sebelum penghapusan: " + warna);

// Menghapus warna yang terdapat pada objek warnaDihapus

warna.removeAll(warnaDihapus);

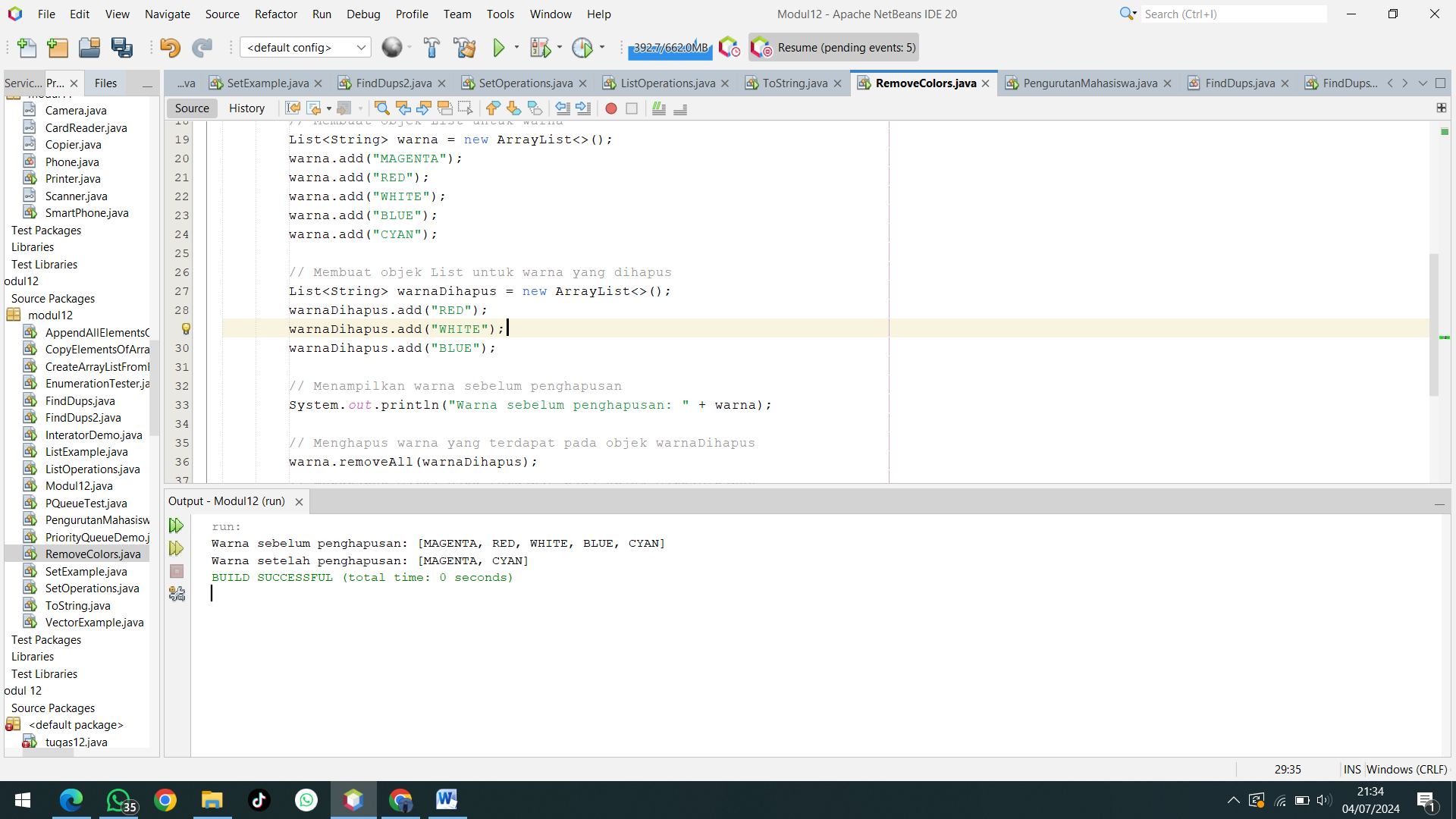
// Menampilkan warna setelah penghapusan

System.out.println("Warna setelah penghapusan: " + warna);

}

}

**Output :**



1. Deklarasi dan Inisialisasi List

* warna List:
  + List<String> warna = new ArrayList<>(); mendeklarasikan dan menginisialisasi sebuah ArrayList yang menyimpan elemen bertipe String.
  + warna.add(...) menambahkan beberapa elemen warna ke dalam list, termasuk "MAGENTA", "RED", "WHITE", "BLUE", dan "CYAN".
* warnaDihapus List:
  + List<String> warnaDihapus = new ArrayList<>(); mendeklarasikan dan menginisialisasi sebuah ArrayList untuk menyimpan warna yang akan dihapus.
  + warnaDihapus.add(...) menambahkan elemen warna yang ingin dihapus dari list utama, yaitu "RED", "WHITE", dan "BLUE".

1. Menampilkan Data Sebelum Penghapusan

* System.out.println("Warna sebelum penghapusan: " + warna);: Menampilkan list warna sebelum melakukan operasi penghapusan. Outputnya akan:

**Warna sebelum penghapusan: [MAGENTA, RED, WHITE, BLUE, CYAN]**

1. Menghapus Data dari List Utama

* warna.removeAll(warnaDihapus);: Menghapus semua elemen yang terdapat dalam warnaDihapus dari list warna. Metode removeAll akan menghapus setiap elemen di warna yang ada di dalam warnaDihapus. Setelah operasi ini, elemen "RED", "WHITE", dan "BLUE" akan dihapus dari warna.

1. Menampilkan Data Setelah Penghapusan

* System.out.println("Warna setelah penghapusan: " + warna);: Menampilkan list warna setelah operasi penghapusan. Outputnya akan:

**Warna setelah penghapusan: [MAGENTA, CYAN]**

**Kesimpulan**

* List Sebelum Penghapusan: List warna berisi elemen [MAGENTA, RED, WHITE, BLUE, CYAN].
* Operasi Penghapusan: Elemen-elemen yang ada dalam warnaDihapus ("RED", "WHITE", "BLUE") dihapus dari warna menggunakan removeAll.
* List Setelah Penghapusan: List warna berisi elemen yang tersisa setelah penghapusan, yaitu [MAGENTA, CYAN].
* Penjelasan Metode removeAll
* removeAll(Collection<?> c): Menghapus semua elemen dari list yang ada dalam koleksi yang diberikan (c). Dalam kasus ini, elemen yang ada dalam warnaDihapus dihapus dari warna.
* **Tugas 2**

package modul12;

/\*\*

\*

\* @author Willy

\*/

//TUGAS 2

import java.util.\*;

// Kelas Mahasiswa

class Mahasiswa {

String nrp;

String nama;

float nilai;

// Constructor

public Mahasiswa(String nrp, String nama, float nilai) {

this.nrp = nrp;

this.nama = nama;

this.nilai = nilai;

}

// Getter untuk nilai

public float getNilai() {

return nilai;

}

@Override

public String toString() {

return "NRP: " + nrp + ", Nama: " + nama + ", Nilai: " + nilai;

}

}

public class PengurutanMahasiswa {

public static void main(String[] args) {

// Membuat queue untuk menyimpan data mahasiswa

Queue<Mahasiswa> queue = new LinkedList<>();

// Menambahkan 10 data mahasiswa ke dalam queue

for (int i = 1; i <= 10; i++) {

queue.add(new Mahasiswa(String.format("NRP%03d", i), "Mahasiswa " + i, generateRandomNilai()));

}

// Mengubah queue menjadi list untuk pengurutan

List<Mahasiswa> listMahasiswa = new ArrayList<>(queue);

// Mengurutkan list berdasarkan nilai secara menurun

listMahasiswa.sort(Comparator.comparing(Mahasiswa::getNilai).reversed());

// Menampilkan hasil pengurutan

System.out.println("Data Mahasiswa setelah diurutkan berdasarkan nilai:");

for (Mahasiswa mhs : listMahasiswa) {

System.out.println(mhs);

}

}

// Metode untuk menghasilkan nilai random antara 60 hingga 100

private static float generateRandomNilai() {

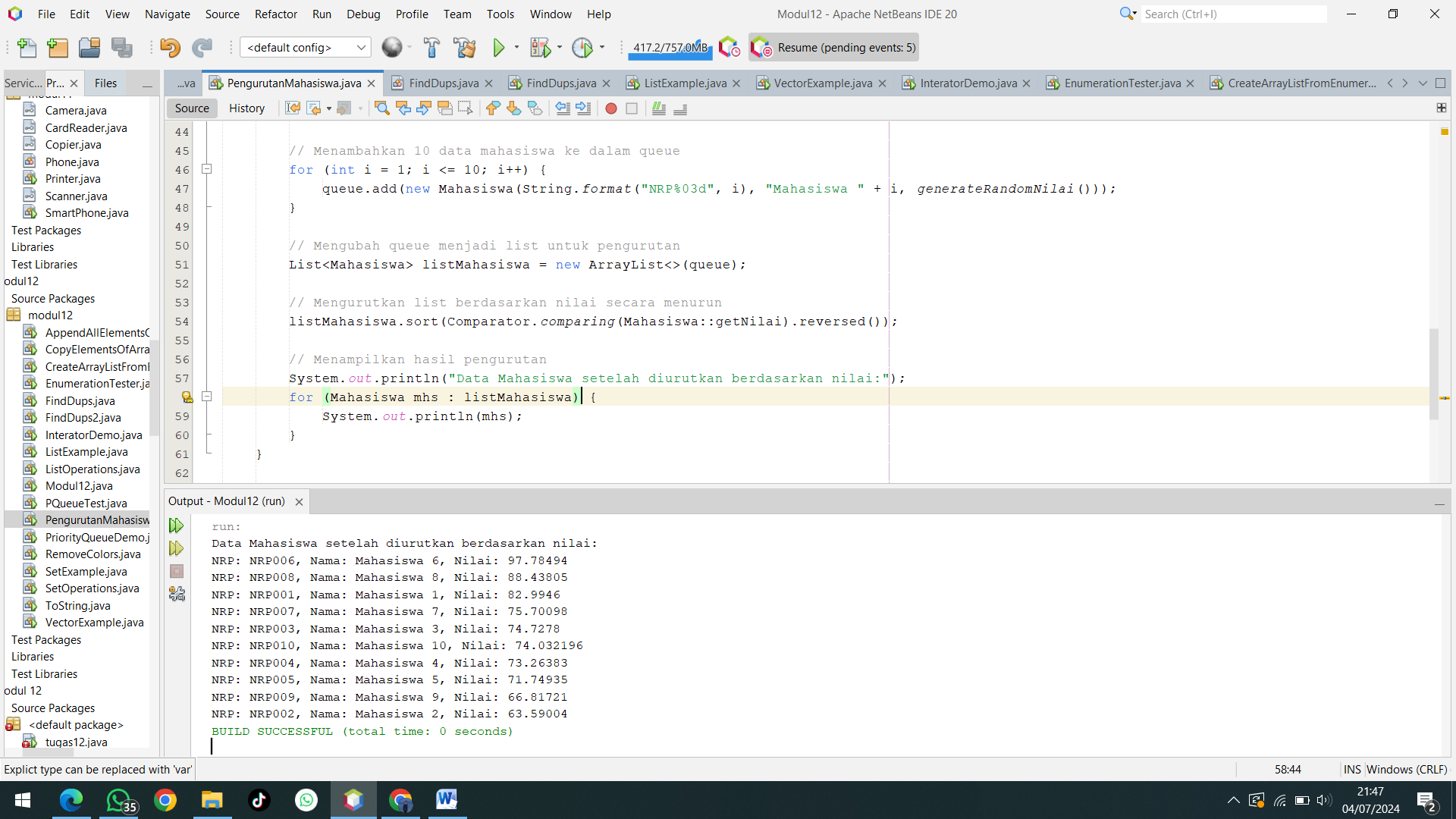
Random rand = new Random();

return 60 + rand.nextFloat() \* 40; // Nilai antara 60 dan 100

}

}

**Output :**



1. Kelas Mahasiswa

* Atribut:
  + nrp (String): Nomor Registrasi Peserta.
  + nama (String): Nama mahasiswa.
  + nilai (float): Nilai mahasiswa.
* Constructor: Mahasiswa(String nrp, String nama, float nilai) digunakan untuk menginisialisasi objek Mahasiswa dengan nrp, nama, dan nilai.
* Getter: getNilai() mengembalikan nilai mahasiswa.
* Metode toString: Mengoverride metode toString untuk memberikan representasi string yang jelas dari objek Mahasiswa, yang mencakup nrp, nama, dan nilai.

1. Metode generateRandomNilai

* Menghasilkan nilai random antara 60 dan 100 menggunakan Random. Metode ini digunakan untuk mengisi nilai mahasiswa secara acak.

1. Kelas PengurutanMahasiswa

* Queue untuk Mahasiswa:
  + Queue<Mahasiswa> queue = new LinkedList<>();: Mendeklarasikan dan menginisialisasi objek Queue yang menyimpan elemen bertipe Mahasiswa. LinkedList adalah implementasi dari Queue di Java.
* Menambahkan Data Mahasiswa:
  + Menggunakan loop for untuk menambahkan 10 objek Mahasiswa ke dalam queue. generateRandomNilai() dipanggil untuk memberikan nilai acak pada setiap mahasiswa.
* Mengubah Queue menjadi List:
  + List<Mahasiswa> listMahasiswa = new ArrayList<>(queue);: Mengkonversi queue menjadi ArrayList agar bisa diurutkan.
* Mengurutkan List Berdasarkan Nilai:
  + listMahasiswa.sort(Comparator.comparing(Mahasiswa::getNilai).reversed());: Mengurutkan listMahasiswa berdasarkan nilai secara menurun (dari yang tertinggi ke yang terendah) menggunakan Comparator.
* Menampilkan Data Mahasiswa:
  + for (Mahasiswa mhs : listMahasiswa) { System.out.println(mhs); }: Menampilkan data mahasiswa yang telah diurutkan.

**Kesimpulan**

* Penggunaan Queue: Queue digunakan untuk menyimpan data mahasiswa secara sementara sebelum diubah menjadi list untuk diurutkan.
* Konversi ke List: Konversi queue menjadi ArrayList untuk memungkinkan pengurutan, karena Queue tidak mendukung operasi pengurutan secara langsung.
* Pengurutan: Pengurutan dilakukan berdasarkan nilai mahasiswa secara menurun menggunakan Comparator.comparing().reversed().
* Pengacakan Nilai: generateRandomNilai() digunakan untuk memberikan nilai acak mahasiswa dalam rentang 60 hingga 100.