­

ARINAL HAQ

2110171051

**COLLECTION (SET DAN LIST)**

1. **PERCOBAAN**
2. Memahami penggunaan class-class yang mengimplementasikan interface Set yaitu class HashSet dan class TreeSet
3. Listing program

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);

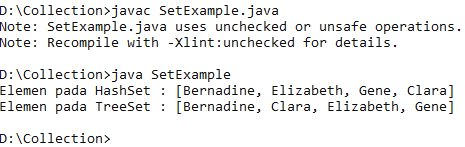
System.out.print("Elemen pada TreeSet : ");

System.out.println(sortSet);

}

}

1. Output program



1. Analisa
2. Penggunaan Class HashSet
3. Listing program

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);

}

}

public class FindDups2 {

public static void main(String[] args) {

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

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

for (String a : args)

if (!uniques.add(a))

dups.add(a);

// Destructive set-difference

uniques.removeAll(dups);

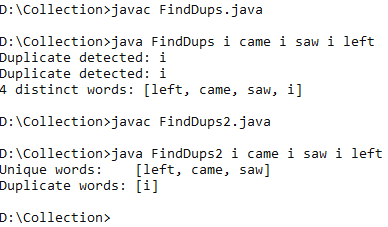
System.out.println("Unique words: " + uniques);

System.out.println("Duplicate words: " + dups);

}

}

1. Output program



1. Analisa
2. Interface Set menerapkan konsep himpunan. Mengetahui implementasi konsep himpunan pada interface Set.
3. Listing program

import java.util.\*;

public class SetExample2 {

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()) {

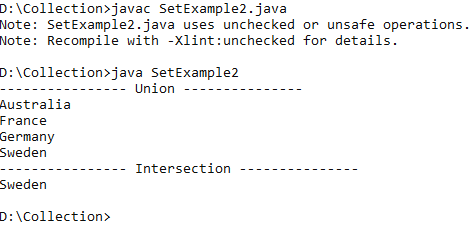
System.out.println(it.next());

}

}

}

1. Output program



1. Analisa
2. Memahami penggunaan class-class yang mengimplementasikan interface List yaitu ArrayList dan LinkedList.
3. Listing program

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();

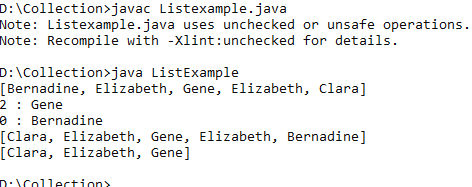
queue.removeLast();

System.out.println(queue);

}

}

1. Output program



1. Analisa
2. Penggunaan Class Vector
3. Listing program

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++) {

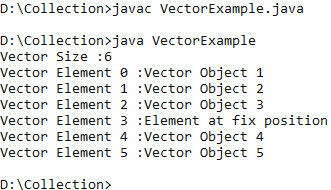
System.out.println("Vector Element " + i + " :" + vc.get(i));

}

}

}

1. Output program



1. Analisa
2. Penggunaan Iterator
3. Listing program

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 + " ");

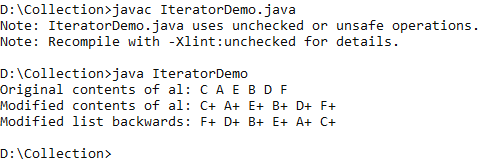
}

System.out.println();

}

}

1. Output program



1. Analisa
2. Penggunaan Enumeration
3. Listing program

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();

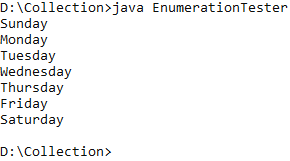
while (days.hasMoreElements())

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

}

}

1. Output program



1. Analisa
2. Membuat Array List dari Enumerasi
3. Listing program

import java.util.\*;

import java.util.Enumeration;

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

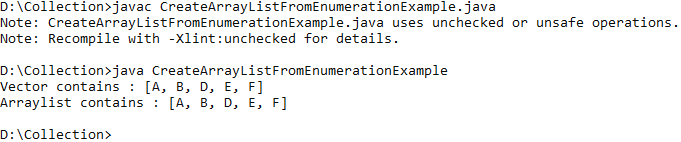
ArrayList aList = Collections.list(e);

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

}

}

1. Output program



1. Analisa
2. Mengkopikan element dari ArrayList ke Vector
3. Listing program

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 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 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);

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

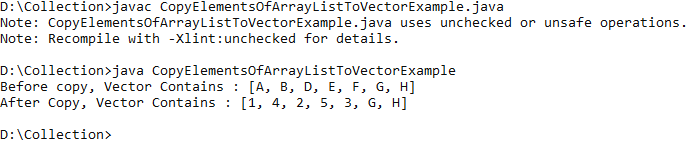
Collections.copy(v,arrayList);

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

}

}

1. Output program



1. Analisa
2. Menambahkan elemen yang tersimpan di Collection pada ArrayList
3. Listing program

import java.util.ArrayList;

public class AppendAllElementsOfOtherCollectionToArrayListExample {

public static void main(String[] args) {

//create an ArrayList object

ArrayList arrayList = new ArrayList();

//Add elements to Arraylist

arrayList.add("1");

arrayList.add("2");

arrayList.add("3");

//create a new Vector object

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

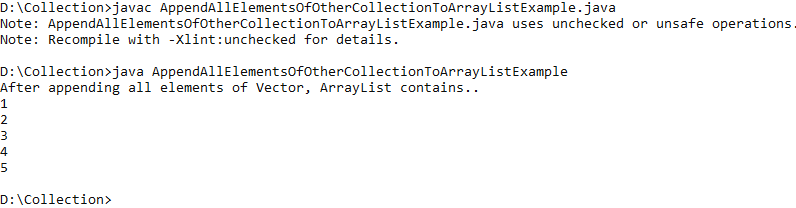
System.out.println(arrayList.get(i));

}

}

}

1. Output program



1. Analisa
2. Memahami Penggunaan dari class PriorityQueue
3. Listing program

import java.util.\*;

public class PriorityQueueDemo {

static PriorityQueue<String> stringQueue;

public static void main(String[] args) {

stringQueue = new PriorityQueue<String>();

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

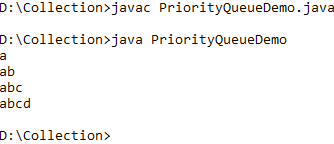
while (stringQueue.size() > 0)

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

}

}

1. Output program



1. Analisa
2. Memahami Penggunaan dari class PriorityQueue dan data yang tersimpan dalam objek PriorityQueue mengimplementasikan interface Comparator.
3. Listing program

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>() {

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 if (flag2) {

return 1;

}

return 0;

}

});

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 (true) {

Integer head = pQueue.poll();

if (head == null) {

break;

}

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

}

}

/\*\*

\*

\* @param n

\* @return

\*/

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;

}

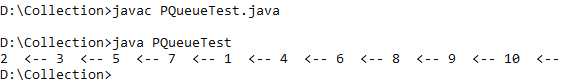
}

return true;

}

}

1. Output program



1. Analisa
2. **LATIHAN**
3. Penerapan konsep himpunan pada interface Set

Terdapat sebuah himpunan

A = {1,2,3,4,5}

B = {5,6,7,8,9,10}

Menggunakan class yang mengimplementasikan Interface Set, dapatkah output seperti :

* A – B
* A ∩ B
* A U B
* A C B

Jawab :

1. Listing program

import java.util.HashSet;

import java.util.Set;

public class Latihan1 {

public static void main(String[] args) {

Set a = new HashSet();

Set b = new HashSet();

int i;

for (i = 1; i < 6; i++) {

a.add(i);

}

for (i = 5; i < 11; i++) {

b.add(i);

}

Set union = new HashSet<>(a);

union.addAll(b);

System.out.println("A gabungan B = " + union);

Set irisan = new HashSet<>(a);

irisan.retainAll(b);

System.out.println("A irisan B = " + irisan);

Set difference = new HashSet<>(a);

difference.removeAll(b);

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

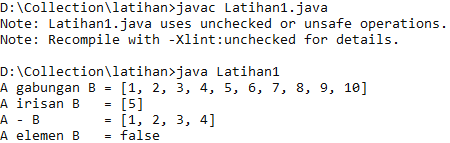
Set elemen = new HashSet<>(a);

System.out.println("A elemen B = " + elemen.containsAll(b));

}

}

1. Output program



1. Analisa
2. Memahami penggunaan interface List.

* Buatlah objek List, dengan data bertipe String lakukan langkah berikut :
* Tampilkan data yang terdapat pada list.
* Baliklah data yang terdapat pada list dan tampilkan.
* Acaklah data tersebut dan tampilkan.
* Urutkan data tersebut dan tampilkan.

Jawab :

1. Listing program

import java.util.\*;

public class Latihan2 {

public static void main(String[] args) {

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

l.add("apel");

l.add("jeruk");

l.add("mangga");

l.add("anggur");

l.add("semangka");

l.add("pisang");

System.out.print("List asli : ");

System.out.println(l);

Collections.reverse(l);

System.out.print("List reverse : ");

System.out.println(l);

Collections.shuffle(l);

System.out.print("List acak : ");

System.out.println(l);

Collections.sort(l);

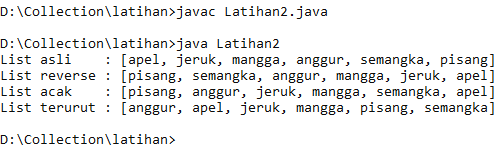
System.out.print("List terurut : ");

System.out.println(l);

}

}

1. Output program



1. Analisa
2. Memahami penggunaan interface List (2)

Buatlah class Mahasiswa dengan informasi nrp dan nama(bertipe String).

Buatlah objek List, dengan data bertipe String lakukan langkah berikut :

* Tampilkan data yang terdapat pada list.
* Baliklah data yang terdapat pada list dan tampilkan.
* Acaklah data tersebut dan tampilkan.
* Urutkan data tersebut, jangan lupa untuk mengimplementasikan interface Comparable/Comparator pada class Mahasiswa dan tampilkan..

Jawab :

1. Listing program

import java.util.\*;

public class Latihan2 {

main(String[] args) {

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

l.add("apel");

l.add("jeruk");

l.add("mangga");

l.add("anggur");

l.add("semangka");

l.add("pisang");

System.out.print("List asli : ");

System.out.println(l);

Collections.reverse(l);

System.out.print("List reverse : ");

System.out.println(l);

Collections.shuffle(l);

System.out.print("List acak : ");

System.out.println(l);

Collections.sort(l);

System.out.print("List terurut : ");

System.out.println(l);

}

}

1. Output program
2. Analisa
3. **TUGAS**
   1. Buatlah sebuah class Stack, FullStackException dan EmptyStackException. Class Stack ini menggambar Stack yang menerapkan konsep LIFO (Last In First Out). Konsep LIFO ini, data yang terakhir masuk akan keluar pertama kali
4. Listing program

* Class Stack

public class Stack {

int size;

int top;

Object[] elemen;

public Stack() {

this.size = 5;

this.elemen = new Object[size];

}

public Stack(int s) {

this.size = s;

this.elemen = new Object[size];

}

public int getSize() {

return size;

}

public int getTop() {

return top;

}

public void Push(Object o) throws FullStackException {

if(top==size){

throw new FullStackException("Stack Penuh");

}else{

this.elemen[top] = o;

top = top + 1;

}

}

public Object Pop() throws EmptyStackException {

if(top==0){

throw new EmptyStackException("Stack Kosong");

}

else{

top--;

Object x = elemen[top];

return x;

}

}

}

public class FullStackException extends RuntimeException{

public FullStackException(String s) {

super(s);

}

}

public class EmptyStackException extends RuntimeException{

public EmptyStackException(String s) {

super(s);

}

}

public class Test {

public static void main(String[] args) {

Stack st = new Stack();

try {

st.Push("Satu");

st.Push("Dua");

st.Push("Tiga");

st.Push("Empat");

st.Push("Lima");

st.Push("Enam");

} catch (FullStackException e) {

System.out.println(e);

}

try {

System.out.println(st.Pop());

System.out.println(st.Pop());

System.out.println(st.Pop());

System.out.println(st.Pop());

System.out.println(st.Pop());

System.out.println(st.Pop());

System.out.println(st.Pop());

} catch (EmptyStackException e) {

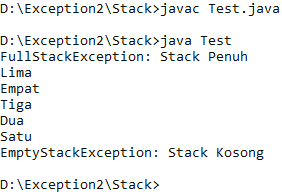
System.out.println(e);

}

}

}

1. Output program



1. Analisa

Program diatas adalah program implementasi exception pada Stack, dimana exception FullStackException dan EmptyStackException dibuat sendiri dengan inherit pada RuntimeException dan constructornya memiliki 1 parameter string untuk menampilkan pesan error tambahan, kemudian di Stack di method Push apabila top sudah mencapai size maka melempar FullStackException, dan di method Pop apabila top sudah 0 maka melempar EmptyStackException, kemudian di main di test method pop dan push diletakkan di blok try kemudian di catch dengan masing-masing exception apabila terjadi exception