LABORATORIUM PEMBELAJARAN ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

BAB	: ADT ARRAY 1 DIMENSI
NAMA	: CHARISMA PRASETYA PUTERA AMMAL
NIM	: 185150700111019
TANGGAL	: 13/09/2019
ASISTEN	: AFIF MUSYAYYIDIN

A. DEFINISI MASALAH

1. Kembangkan program di atas dengan menambahkan method berikut :
   a. Mencari posisi bilangan yang merupakan kelipatan dari x0 yang terletak diantara index x1 dan index x2 dengan header sebagai berikut :

      int findPosKelipatan (int x0, int x1, int x2)

      Contoh : Sebuah set {1, 2, 3, 4, 5, 6, 7} dan dengan memanggil method findPosKelipatan(3, 1, 6) maka akan menghasilkan 2 5 karena kelipatan dari nilai 3 adalah 3 dan 6, dimana keduanya ada di antara index 1 – 6 dan memiliki indeks 2 dan indeks 5.

   b. Pengurutan dengan menggunakan algoritma pengurutan yang lain selain Selection Sort
2. Pada Latihan kedua ini anda diminta untuk melengkapi bagian dari program ADT_Larik sehingga jika diberikan program utama pada gambar 1 akan menghasilkan keluaran sebagaimana gambar 2

Program Latihan Praktikum 2.2

```
1   package ADT_Larik;
2
3   /**
4    *
5    * @author ASUS
6    */
7   public class Larik {
8   //data (strukur data)
9
10      private int size;
11      private double[] itemDt;
12
13      /**
14       * Contructor untuk membuat ADT larik dari suatu
    array
15       *
16       *
17       * @param A : array bertipe int
18       */
19      public Larik(double[] A) {
20          this.size = A.length;
21          this.itemDt = new double[this.size];
22          for (int i = 0; i < this.size; i++) {
23              this.itemDt[i] = A[i];
24          }
25      }
```

```
26
27   /**
28        * fungsi untu mendapatkan ukuran larik
29        *
30        * @return size dari larik
31        */
32       public int getSize() {
33           return this.size;
34       }
35
36       /**
37        * fungsi untuk mendapatkan item ke i dari suatu
     larik
38        *
39        * @param i : posisi item
40        * @return item larik
41        */
42       public double getItem(int i) {
43           return this.itemDt[i];
44       }
45
46       /**
47        * fungsi static untuk menyambung dua buah larik l1
     dan l2
48        *
49        * @param l1 : Larik
50        * @param l2 : Larik
51        * @return Larik
52        */
53       public static Larik sambung(Larik l1, Larik l2) {
54   // Lengkapi bagian ini
55       }
56
57       /**
58        * procedure untuk isiItem suatu larik
59        *
60        * @param id : indeks larik
61        * @param dt : item data yang akan disisipkan
62        */
63       public void isiItem(int id, double dt) {
64           this.itemDt[id] = dt;
65       }
66
67       /**
68        * procedure cetak suatu array
69        *
70        * @param komentar : String
71        */
72       public void cetak(String komentar) {
73           System.out.println(komentar);
74           for (int i = 0; i < this.size; i++) {
75               System.out.printf("%.2f ", this.itemDt[i]);
76           }
77           System.out.println();
78       }
79
80       /**
81        * fungsi untuk mendapatkan nilai terbesar dari
     suatu larik
```

```
 82      *
 83      * @return : item tebesar dari larik
 84      */
 85     public double findBesar() {
 86         double besar = this.itemDt[0];
 87         for (int i = 1; i < this.size; i++) {
 88             if (besar < this.itemDt[i]) {
 89                 besar = this.itemDt[i];
 90             }
 91         }
 92         return besar;
 93     }
 94 
 95     /**
 96      * fungsi untuk mencari posisi suatu data tertentu
    di array
 97      *
 98      * @param dtCari : data yang akan dicari
 99      * @return posisiData
100      */
101     public int getPosisi(double dtCari) {
102         int pos = -99;
103         boolean ketemu = false;
104         int i = 0;
105         while (!ketemu && i < this.size) {
106             if (dtCari == this.itemDt[i]) {
107                 ketemu = true;
108                 pos = i;
109             }
110             i++;
111         }
112         return pos;
113     }
114 
115     /**
116      * fungsi static untuk mencopy isi suatu larik l
117      *
118      * @param k : posisi awal
119      * @param n : jumlah item yang akan dicopy
120      * @param l : larik asal
121      * @return Larik hasil copy
122      */
123     public static Larik copyLarik(int k, int n, Larik l)
    {
124 // lenkapi bagian ini
125     }
126 
127     /**
128      * fungsi untuk mencari posisi terbesar suatu data
    suatu posisi awal sampai
129      * akhir
130      *
131      * @param awal : posisi awal
132      * @param akhir : posisi akhir
133      * @return posisi data terbesar
134      */
135     public int getPosBesar(int awal, int akhir) {
136         int posBesar = -1;
137         double itemBesar;
```

```
138         if (awal <= akhir) {
139             posBesar = awal;
140             itemBesar = this.getItem(awal);
141 
142             for (int i = awal + 1; i < akhir; i++) {
143                 double nilaiItem = this.getItem(i);
144                 if (itemBesar < nilaiItem) {
145                     itemBesar = nilaiItem;
146                     posBesar = i;
147                 }
148             }
149         }
150         return posBesar;
151     }
152 
153     /**
154      * fungsi untuk mencari posisi data terkecil suatu
    array mulai dari posisi
155      * awal sampai posisi akhir
156      *
157      * @param awal : posisi awal
158      * @param akhir : posisi akhir
159      * @return posisi data terkecil
160      */
161     public int getPosKecil(int awal, int akhir) {
162 // lenkapi bagian ini
163     }
164 
165     /**
166      * fungsi pengurutan suatu larik lAsal dimana
    kondisi lAsal akan tetap
167      * setelah proses pengurutan
168      *
169      * @param lAsal : Array asal yang akan diurutkan
170      * @param status : 0-> urut dari kecil ke besar 1->
    urut dari besar ke kecil
171      * @return Array baru hasil pengurutan
172      */
173     public static Larik SelectionSort(Larik lAsal, int
    status) {
174         int n = lAsal.getSize();
175         Larik lhasil = Larik.copyLarik(0, n, lAsal);
176 
177         if (status == 0) {// urutkan data dari kecil ke
    besar
178             for (int i = 0; i < n; i++) {
179                 int posKecil = lhasil.getPosKecil(i, n);
180                 double itemKecil  = hasil.getItem
    (posKecil);
181                 double itemI = lhasil.getItem(i);
182                 lhasil.isiItem(i, itemKecil);
183                 lhasil.isiItem(posKecil, itemI);
184             }
185         } else { // urutkan data dari besar ke kecil
186             for (int i = 0; i < n; i++) {
187                 int posBesar = lhasil.getPosBesar(i, n);
188                 double itemBesar =
    lhasil.getItem(posBesar);
189                 double itemI = lhasil.getItem(i);
```

```
                    lhasil.isiItem(i, itemBesar);
                    lhasil.isiItem(posBesar, itemI);
                }
            }
            return lhasil;
        }
}
```

Gambar 2. Hasil keluaran

```
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
Isi Larik L2
4.00 3.00 1.00 11.00 7.00
L3 = gabungan dari L1 dan L2
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00 4.00 3.00 1.00 11.00
7.00
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
L4 Copy dari L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
L5 Hasil pengurutan dari L1 kecil -> besar
1.00 2.00 3.00 4.00 5.00 10.00 10.00 16.00 20.00
L6 Hasil pengurutan dari L1 besar -> kecil
20.00 16.00 10.00 10.00 5.00 4.00 3.00 2.00 1.00
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
HASIL KALI Larik L1*L4 = 911.000
```

B. SOURCE CODE

namaClass.java

```
public class Larik2 {
    private int size;
    private int[] itemDt;
    public void buatLarik(int n) {
        this.size = n;
        this.itemDt = new int[this.size];
    }
    public Larik2(int n) {
        buatLarik(n);
    }
    public int getSize() {
        return this.size;
    }
    public Larik2(int[] dt) {
        buatLarik(dt.length);
        for (int i = 0; i < dt.length; i++) {
            isiItem(i, dt[i]);
        }
    }
    public void isiItem(int id, int dt) {
        this.itemDt[id] = dt;
    }
    public void cetak(String komentar) {
        System.out.println(komentar);
        for (int i = 0; i < this.size; i++) {
            System.out.print(this.itemDt[i] + " ");
        }
        System.out.println();
    }
    public Larik2 copyLarik(int k, int n) {
        Larik2 lHasil = null;
```

```
32          if (n <= this.size - k) {
33              lHasil = new Larik2(n);
34              int j = 0;
35              for (int i = k; i < k + n; i++) {
36                  lHasil.isiItem(j++, this.itemDt[i]);
37              }
38          }
39          return lHasil;
40      }
41      public Larik2 BubbleSort(int pilihan) {
42          Larik2 bubble = copyLarik(0, size);
43          for (int a = 0; a < bubble.getSize(); a++) {
44            for (int b = 0; b < bubble.getSize() - 1; b++)
    {
45               if (bubble.itemDt[b] > bubble.itemDt[b +
    1]) {
46                     int temp = bubble.itemDt[b];
47                   bubble.itemDt[b] = bubble.itemDt[b +
    1];
48                     bubble.itemDt[b + 1] = temp;
49                }
50              }
51          }
52          return bubble;
53      }
54      public int FindKelipatan(int x0, int x1, int x2) {
55          for (int x = x1; x <= x2; x++) {
56              if (this.itemDt[x] % x0 == 0) {
57                  System.out.println(x);
58              }
59          }
60          return 0;
61      }
62      public static void main(String[] args) {
63          int[] A = {2, 34, 8, 7, 10};
64          Larik2 lA = new Larik2(A);
65          System.out.println("Pos kelipatan");
66          lA.FindKelipatan(2, 0, 4);
67          lA.BubbleSort(0).cetak("sesudah di bubble");
68      }
69  }
```

Larik.java

```
1   package ADT_Larik;
2   public class Larik {
3       private int size;
4       private double[] itemDt;
5       public Larik(double[] A) {
6           this.size = A.length;
7           this.itemDt = new double[this.size];
8           for (int i = 0; i < this.size; i++) {
9               this.itemDt[i] = A[i];
10          }
11      }
12      public int getSize() {
13          return this.size;
```

```
14      }
15      public double getItem(int i) {
16          return this.itemDt[i];
17      }
18      public static Larik sambung(Larik l1, Larik l2) {
19          int indek = 0;
20          double sambungArray[] = new double[l1.size +
    l2.size];
21          Larik sambungan;
22          for (int i = 0; i < l1.size; i++) {
23              sambungArray[i] = l1.itemDt[i];
24          }
25          for (int i = l1.size; i < sambungArray.length;
    i++) {
26              sambungArray[i] = l2.itemDt[indek];
27              indek++;
28          }
29          sambungan = new Larik(sambungArray);
30          return sambungan;
31
32      }
33      public void isiItem(int id, double dt) {
34          this.itemDt[id] = dt;
35      }
36      public void cetak(String komentar) {
37          System.out.println(komentar);
38          for (int i = 0; i < this.size; i++) {
39              System.out.printf("%.2f ", this.itemDt[i]);
40          }
41          System.out.println();
42      }
43      public double findBesar() {
44          double besar = this.itemDt[0];
45          for (int i = 1; i < this.size; i++) {
46              if (besar < this.itemDt[i]) {
47                  besar = this.itemDt[i];
48              }
49          }
50          return besar;
51      }
52      public int getPosisi(double dtCari) {
53          int pos = -99;
54          boolean ketemu = false;
55          int i = 0;
56          while (!ketemu && i < this.size) {
57              if (dtCari == this.itemDt[i]) {
58                  ketemu = true;
59                  pos = i;
60              }
61              i++;
62          }
63          return pos;
64      }
65      public static Larik copyLarik(int k, int n, Larik l)
    {
66          double copy[] = new double[n];
67          Larik a;
68          for (int i = k; i < k + n; i++) {
69              copy[i] = l.itemDt[i];
```

```
70          }
71          a = new Larik(copy);
72          return a;
73
74      }
75      public int getPosBesar(int awal, int akhir) {
76          int posBesar = -1;
77          double itemBesar;
78          if (awal <= akhir) {
79              posBesar = awal;
80              itemBesar = this.getItem(awal);
81
82              for (int i = awal + 1; i < akhir; i++) {
83                  double nilaiItem = this.getItem(i);
84                  if (itemBesar < nilaiItem) {
85                      itemBesar = nilaiItem;
86                      posBesar = i;
87                  }
88              }
89          }
90          return posBesar;
91      }
92      public int getPosKecil(int awal, int akhir) {
93          int posKecil = -1;
94          double kecil;
95          if (awal < akhir) {
96              kecil = this.getItem(awal);
97              for (int i = awal; i < akhir; i++) {
98                  if (this.itemDt[i] < kecil) {
99                      kecil = itemDt[i];
100                 }
101             }
102             posKecil = getPosisi(kecil);
103         }
104         return posKecil;
105     }
106     public static double LarikKaliLarik(Larik L1, Larik
    L4) {
107         double hasil = 0;
108         if (L1.itemDt.length == L4.itemDt.length) {
109             for (int i = 0; i < L1.size; i++) {
110                 hasil += L1.itemDt[i] * L4.itemDt[i];
111             }
112         }
113         return hasil;
114     }
115     public static Larik SelectionSort(Larik lAsal, int
    status) {
116         int n = lAsal.getSize();
117         Larik lhasil = Larik.copyLarik(0, n, lAsal);
118
119         if (status == 0) {
120             for (int i = 0; i < n; i++) {
121                 int posKecil = lhasil.getPosKecil(i, n);
122                 double itemKecil =
    lhasil.getItem(posKecil);
123                 double itemI = lhasil.getItem(i);
124                 lhasil.isiItem(i, itemKecil);
125                 lhasil.isiItem(posKecil, itemI);
```

```
126             }
127         } else {
128             for (int i = 0; i < n; i++) {
129                 int posBesar = lhasil.getPosBesar(i, n);
130                 double itemBesar =
    lhasil.getItem(posBesar);
131                 double itemI = lhasil.getItem(i);
132                 lhasil.isiItem(i, itemBesar);
133                 lhasil.isiItem(posBesar, itemI);
134             }
135         }
136         return lhasil;
137     }
138 }
```

AppPr1.java

```
 1 package ADT_Larik;
 2 public class AppPr1 {
 3     public static void main(String[] args) {
 4         double[] A = {3, 4, 1, 10, 5, 2, 10, 20, 16};
 5         double[] B = {4, 3, 1, 11, 7};
 6         Larik L1 = new Larik(A);
 7         Larik L2 = new Larik(B);
 8         L1.cetak("Isi Larik L1");
 9         L2.cetak("Isi Larik L2");
10         Larik L3 = Larik.sambung(L1, L2);
11         L3.cetak("L3 = gabungan dari L1 dan L2");
12         Larik L4 = Larik.copyLarik(0, L1.getSize(), L1);
13         L1.cetak("Isi Larik L1");
14         L4.cetak("L4 Copy dari L1");
15         Larik L5 = Larik.SelectionSort(L1, 0);
16         L5.cetak("L5 Hasil pengurutan dari L1 kecil ->
    besar");
17         Larik L6 = Larik.SelectionSort(L1, 1);
18         L6.cetak("L6 Hasil pengurutan dari L1 besar ->
    kecil");
19         L1.cetak("Isi Larik L1");
20         double hasil = Larik.LarikKaliLarik(L1, L4);
21         System.out.printf("HASIL KALI %.3f\n", hasil);
22     }
23 }
```

C. PEMBAHASAN

Larik2.java	
1	Deklarasi nama kelas dengan nama Larik2
2	Membuat variable int dengan nama size
3	Membuat array 1 dimensi dengan nama itemDt
4	Membuat method void buatLarik dengan parameter berisi int n
5	Deklarasi fungsi this size = n
6	Deklarasi fungsi this itemDt = deklarasi array int dengan nilai [this.size]
7	Sintak pentutup method
8	Membuat method Larik2 dengan parameter int n
9	Deklarasi method buatLarik dengan nilai parameter n
10	Sintak penutup method

11	Membuat method getsize
12	Fungsi return dengan kembalian this.size
13	Sintak penutup method
14	Membuat method overloading Larik2 dengan parameter int [] dt
15	Deklarasi method buatLarik dengan parameter dt.lenght
16	Deklarasi fungsi for dengan int i = 0 apabila i kurang dari nilai dt.lenght dan i dilakukan increment
17	Deklarasi void isiItem dengan nilai i dan dt[i]
18	Sintak penutup fungsi for
19	Sintak pentutup method
20	Membuat method void isiItem dengan parameter int id dan int dt
21	Deklarasi fungsi this itemDt[id] = dt
22	Sintak penutup method
23	Membuat method void cetak dengan parameter String komentar
24	Menampilkan keluaran berupa nilai komentar
25	Deklarasi fungsi for int i = 0 , apabila i < this.size maka i akan di increment
26	Menampilkan keluaran berupa nilai this.itenDt [i] + " "
27	Sintak penutup for
28	Menampilkan output kosong dengan fungsi ln
29	Sintak penutup method
30	Membuat method Larik2 dengan nama copyLarik dengan parameter int k dan int n
31	Deklarasi variable Larik2 dengan nama lHasil dengan nilai null
32	Deklarasi fungsi if apabila n kurang dari sama dengan this.size dikurangi k
33	Maka lHasil sama dengan new Larik2 dengan parameter n
34	Membuat variable integer j dengan nilai 0
35	Fungsi for apabila int i = k dan i kurang dari k + n , maka i akan di increment
36	Deklarasi lHasil.isiItem dengan parameter j di increment dan this.itemDt[i]
37	Sintak penutup for
38	Sintak penutup if
39	Fungsi return dengan kembalian lHasil
40	Sintak penutup method
41	Membuat method overloading Larik2 dengan nama Bubblesort dan parameter bernilai int pilihan
42	Deklarasi variable Larik2 bubble dengan nilai copyLarik dan nilai parameter 0,size
43	Fungsi for apabila int a = 0 dan a < bubble.getsize() , maka nilai a akan di increment
44	Fungsi for apabila b = 0 dan b < bubble.getsize() – 1 , maka nilai b akan di increment
45	Deklarasi fungsi if apabila nilai bubble.itemDt[b] > bubble.ite,Dt[b+1]
46	Deklarasi variable integer dengan nama temp dan nilai bubble.itemDt[b]
47	Mengubah nilai bubble.itemDt[b] menjadi bubble.itemDt [b+1]
48	Mengubah nilai bubble.itemDt [b+1] = temp
49	Sintak penutup fungsi if
50	Sintak penutup fungsi for
51	Sintak pentutup fungsi for
52	Fungsi return dengan kembalian bubble
53	Sintak penutup method
54	Membuat method dengan variable int dan nama FindKelipatan yang mempunyai parameter int x0 , int x2 , int x2

55	Deklarasi fungsi for apabila int x = x1 dan x <= x2 maka nilai x akan di increment
56	Deklarasi fungsi if apabila nilai this.itemDt [x] modulu % x0 menghasilkan nilai 0
57	Akan menampilkan ouput nilai x
58	Sintak penutup fungsi if
59	Sintak penutup fungsi for
60	Fungsi return dengan kembalian 0
61	Sintak penutup method
62	Deklarasi fungsi psvm
63	Membuat array integer A dengan nilai 2,34,5,7,10
64	Intansiasi variable Larik2 dengan nama lA
65	Menampilkan keluaran berupa “Pos kelipatan”
66	Memanggil FindKepilapatan dengan parameter 2,0,4 pada variable lA
67	Memanggil BubbleSort 0 dan method cetak “sesudah di bubble” melalui variable lA
68	Sintak penutup fungsi psvm
69	Sintak penutup kelas

Larik.java	
1	Deklarasi package ADT_Larik
2	Deklarasi kelas dengan nama Larik
3	Membua private int dengan nama size
4	Membuat private array double dengan nama itemDt
5	Membuat method Larik dengan parameter double array A
6	Deklarasi fungsi this size bernilai A.lenght
7	Deklarasi fungsi this item.Dt bernilai variable array double [this.size]
8	Fungsi for apabila i = 0 dan i < this.size, maka i akan di increment
9	Fungsi this item[i] bernilai array A[i]
10	Sintak penutup for
11	Sintak penutup method
12	Membuat method integer getsize
13	Fungsi return dengan kembalian this.size
14	Sintak penutup method
15	Membuat method double getItem dengan parameter int i
16	Fungsi return dengan kembalian this.ItemDt[i]
17	Sintak penutup method
18	Membuat method static Larik dengan nama sambung dan parameter Larik l1, Larik L2
19	Membuat variable integer indek bernilai 0
20	Membuat array double dengan nama sambungArray dengan nilai l1.size + 12.size
21	Deklarasi variable Larik dengan sambungan
22	Deklarasi fungsi for apabila i = 0 dan i < 11.size , maka i akan di increment
23	Mengubah nilai sambungArray[i] menjadi l1.itemDt[i]
24	Sintak penutup for
25	Fungsi for apabila int i = 11.size dan i < sambungArray.lenght, maka i akan di increment
26	Mengubah nilai sambungArray[i] menjadi l2.itemDt[indek]
27	Fungsi increment indek

28	Sintak penutup for
29	Deklarasi variable sambungan dengan nilai method Larik dan parameter sambung Array
30	Fungsi return dengan kembalian sambungan;
31	
32	Sintak penutup method
33	Membuat method void dengan nama isiItem dan parameter int id , double dt
34	Fungsi this.itemDt[id] = dt
35	Sintak penutup method
36	Membuat method void dengan nama cetak dan parameter String komentar
37	Menampilkan keluaran variable string komentar
38	Fungsi for apabila int i = 0 dan i < this.size maka i akan di increment
39	Menampilkan keluaran berupa “%.2f ” dan this.itemDt[i]
40	Sintak penutup for
41	Menampilkan output dengan fungsi ln
42	Sintak penutup method
43	Membuat method double dengan nama findBesar
44	Mmebuat variable besar dengan nilai this.itemDt[0]
45	Deklarasi fungsi for apabila int i = 1 dan i < this.size maka i akan di increment
46	Deklarasi fungsi if apabila besar < this.itemDt[i]
47	Maka nilai besar akan menjadi this.itemDt[i]
48	Sintak penutup if
49	Sintak penutup for
50	Fungsi return dengan kembalian besar
51	Sintak penutup method
52	Membuat method int dengan nama getPosisi dan parameter double dtCari
53	Membuat variable int dengan nama pos yang bernilai -99
54	Membuat variable boolean dengan nama ketemu dan bernilai false
55	Membuat variable int dengan nama i dan bernilai 0
56	Deklarasi fungsi while ketika !ketemu dan i < this.size
57	Deklarasi fungsi if apabila dtCari == this.itemDt [i]
58	Maka nikai ketemu berubah menjadi true
59	Nilai pos bernilai i
60	Sintak penutup if
61	Fungsi increment pada i
62	Sintak penutup while
63	Fungsi return dengan kembalian pos
64	Sintak penutup method
65	Membuat method static Larik dengan nama copyLarik dan parameter int k , int n , Larik l
66	Membuat array double dengan nama copy yang bernilai double [n]
67	Membuat variable Larik dengan nama a
68	Fungsi for apabila int i = k dan i < k + n maka i akan di increment
69	Mengisi nilai copy [i] dengan l.itemDt[i]
70	Sintak penutup for
71	Dekalrasi nilai a dengan method Larik(copy)
72	Fungsi return dengan kembalian a
73	
74	Sintak penutup method
75	Membuat method integer dengan nama getPosBesar dan parameter int awal, int akhir

76	Membuat variable int dengan nama posBesar dengan nilai -1
77	Membuat variable double itemBesar
78	Fungsi if apabila nilai awal <= akhir
79	Deklarasi nilai posBesar menjadi awal
80	Deklarasi nilai itemBesar menjadi this.getItem dengan parameter awal
81	
82	Membuat fungsi for apabila int i = awal + 1 dan i < akhir maka i akan di increment
83	Membuat variabel nilaiItem dengan nilai this.getItem dengan parameter awal
84	Membuat fungsi if apabila itemBesar < nilaiItem
85	Deklarasi nilai itemBesar bernilai nilaiItem
86	Deklarasi nilai posBesar dengan nilai variable i
87	Sintak penutup if
88	Sintak penutup for
89	Sintak penutup if
90	Fungsi return dengan kembalian nilai posBesar
91	Sintak penutup method
92	Membuat method integer dengan nama getPosKecil dan parameter int awal, int akhir
93	Membuat variable int posKecil dengan nilai -1
94	Membuat variable double kecil
95	Deklarasi fungsi if apabila awal < akhir
96	Deklarasi nilai kecil dengan this.getItem dengan parameter awal
97	Membuat fungsi for apabila int i bernilai awal dan i < akhir maka i akan di increment
98	Membuat fungsi if apabila this.itemDt[i] < kecil
99	Deklarasi nilai variable kecil dengan itemDt[i]
100	Sintak penutup if
101	Sintak penutup for
102	Deklarasi nilai posKecil dengan getPosisi dengan parameter kecil
103	Sintak penutup method
104	Fungsi return dengan kembalian nilai posKecil
105	Sintak penutup method
106	Membuat method static double dengan nama LarikKaliLarik dan parameter Larik L1, Larik L4
107	Membuat variable double dengan nama hasil dan nilai 0
108	Membuat fungsi if apabila L1.itemDt.length sama dengan L4.itemDt.lenght
109	Membuat fungsi for apabila int i bernilai 0 ,dan i < L1.size maka i akan di increment
110	Deklarasi nilai hasil ditambah sama dengan L1.itemDt[i] dikali L4.itemDt[i]
111	Sintak penutup for
112	Sintak penutup if
113	Membuat fungsi return dengan kembalian nilai hasil
114	Sintak penutup method
115	Membuat method static Larik dengan nama SelecionSort dan parameter Larik lAsal, int status
116	Membuat variable n dengan nilai lAsal.getSize
117	Membuat variable Larik dengan nama lhasil dan nilai Larik.copyLarik dengan parameter 0, n, lAsal
118	

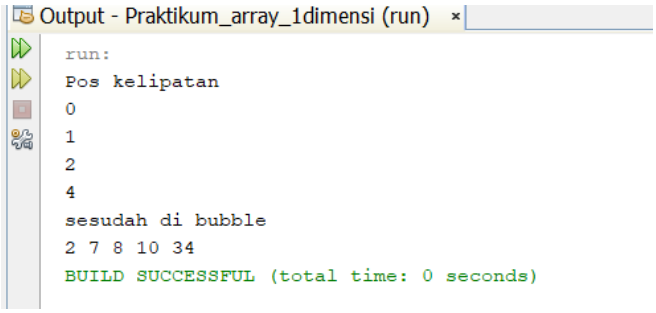
119	Membuat fungsi if apabila status sama dengan 0
120	Membuat fungsi for apabila int i bernilai 0 dan i < n maka i akan di increment
121	Membuat variable int dengan nama posKecil dan nilai lhasil.getPosKecil dengan parameter i, n
122	Membuat variable double dengan nama itemKecil dan nilai lhasil.getItem dengan parameter posKecil
123	Membuat variable double dengan nama itemI dan nilai lhasil.getItem dengan parameter i
124	Memanggil method isiItem dengan parameter i , itemKecil dengan variable lhasil
125	Memanggil method isiItem dengan parameter posKecil , itemI
126	Sintak penutup for
127	Fungsi else dan sintak pembuka
128	Membuat fungsi for apabila int i bernilai 0 dan i < n maka i akan di increment
129	Membuat variable int dengan nama posBesar dan nilai lhasil.getPosBesar dengan parameter i , n
130	Membuat variable double itemBesar dengan nilai lhasil.getItem dengan parameer posBesar
131	Membuat variable double itemI dengan nilai lhasil.getitem dan parameter i
132	Memanggil method isiItem dengan parameter i , itemBesar
133	Memanggil method isiItem dengan parameter posBesar , itemI
134	Sintak penutup for
135	Sintak penutup else
136	Fungsi return dengan nilai kembalian lhasil
137	Sintak penutup method
138	Sintak penutup kelas Larik

AppPr1.java	
1	Deklarasi package ADT_Larik
2	Deklarasi nama kelas dengan nama AppPr1
3	Membuat method psvm
4	Membuat variable double array dengan nama A dan nilai array 3, 4, 1, 10, 5, 2, 10, 20 , 16
5	Membuat variable double array dengan nama B dan nilai array 4, 3, 1, 11, 7
6	Intansiasi objek Larik2 dengan variable L1 dan parameter A
7	Intansiasi objek Larik2 dengan variable L2 dan parameter B
8	Memanggil method cetak dengan parameter “Isi Larik L1” dengan variable L1
9	Memanggil method cetak dengan parameter “Isi Larik L2” dengan variable L2
10	Instansiasi variable L3 melalui objek Larik2 dengan nilai method sambung berparameter L1 , L2 yang di panggil melalui Larik2
11	Memanggil method cetak berparameter string “L3 = gabungan dari L1 dan L2 ”
12	Intansiasi variable L4 melalui objek Larik2 yang bernilai method copyLarik dan parameter 0, L1.getSize(), L1 melalui objek Larik2
13	Memanggil method cetak dengan parameter “Isi Larik L1” dengan variable L1

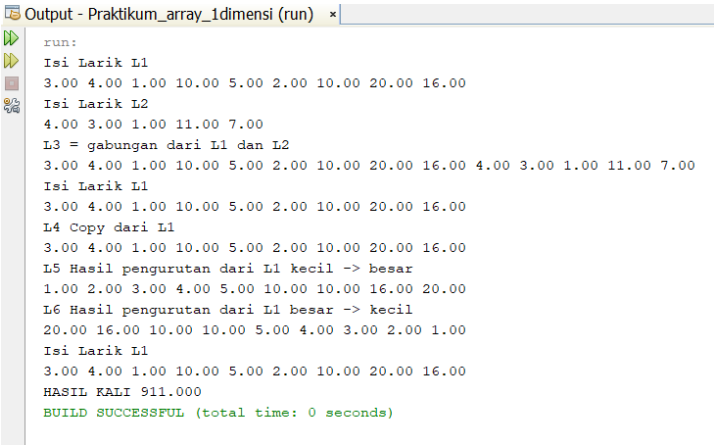
14	Memanggil method cetak dengan parameter “L4 Copy dari L1” dengan variable L4
15	Intansiasi variable L5 melalui objek Larik2 yang bernilai method SelectionSort dan parameter L1, 0 melalui objek Larik2
16	Memanggil method cetak dengan parameter “L5 Hasil pengurutan dari L1 kecil -> besar” dengan variable L5
17	Instansiasi variable L6 melalui objek Larik2 dengan nilai method SelectionSort berparameter L1 , 1 yang di panggil melalui Larik2
18	Memanggil method cetak dengan parameter “L6 Hasil pengurutan dari L1 besar -> kecil” dengan variable L6
19	Memanggil method cetak dengan parameter “Isi Larik L1” dengan variable L1
20	Membuat variable double dengan nama hasil dan nilai method dari LarikKaliLarik berparameter L1, L4 yang di panggil melalui objek Larik2
21	Menampilkan output berupa "HASIL KALI %.3f\n", ditambah variable hasil
22	Sintak penutup psvm
23	Sintak penutup main kelas

D. SCREENSHOT PROGRAM

1.



2.



E. KESIMPULAN

Tipe data abstrak (ADT) dapat didefinisikan sebagai model matematika dari objek data yang menyempurnakan tipe data dengan cara mengaitkannya dengan fungsi-fungsi yang beroprasi pada data yang bersangkutan. Merupakan hal yang

sangat penting untuk mengenali bahwa operasi-operasi yang akan dimanipulasi data pada objek yang bersangkutan termuat dalam spesifikasi ADT

Salah satu contoh penerapan ADT:

```
class Node {

     String Nama;
     Node next;

}
```

Array adalah kumpulandarinilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama Nilai-nilai data disuatu array disebut dengan elemen-elemen array Letak urutan dari elemen-elemen array di tunjukkan oleh suatu subscript atau indek. Cara mendeklarasikan Array :

```
// cara pertama
String[] nama;

// cara kedua
String nama[];

// cara ketiga dengan kata kunci new
String[] nama = new String[5];
```

Cara mengakses array

```
String[] nama = {"Linda", "Santi", "Susan", "Mila", "Ayu"};

// Mengakses array di atas dengan kode program berikut

System.out.println(teman[2]);
```

Macam – macam array

1. Array satu dimensi
   Contoh :
   ```
   int [] umur;
   ```
2. Array dua dimensi
   Contoh :
   ```
   int[][] twoD = new int[512][128]
   ```
3. Array tiga dimensi
   Contoh :
   ```
   char[][][] threeD = new char[8][16][24];
   ```

Penggunaan array 1 dimensi

Penggunaan variabel array dilakukan dengan bentuk:

```
tipe namaVariabelArray[];
```

atau

```
tipe [] namaVariabelArray;
```

Contoh penerapanya :

```
class cobaArray {
   public static void main(String [] args){
   int [] jumlahHari;
   jumlahHari = new int[4];
   jumlahHari[0] = 31;
```

```
        jumlahHari[1] = 28;
        jumlahHari[2] = 31;
        jumlahHari[3] = 30;

        System.out.println("Bulan Maret memiliki " +
jumlahHari[2] + " hari.");
        }
    }
```