|  |  |  |
| --- | --- | --- |
| **LEMBAR KERJA PERSERTA DIDIK ( L K P D )** | | |
| **SMK NEGERI 2 KARANGANYAR** | **Topic:** | **Mata Pelajaran:**  Pemrograman Berorientasi Objek |
| **Kompetensi Keahlian :**  Rekayasa Perangkat Lunak | ***Collection*** | **Kelas/Semester :**  XII/1 |
| **Kompetensi Dasar:**  3.12. Memahami data collection sebagai media penyimpanan data  4.12. Menyajikan data colection sebagai penyimpan data | **Waktu : 16** **Jp @45 menit**  (2 x Pertemuan) |
| **Indikator Pencapaian Kompetensi:**  3.12.2 Menerangkan prosedur *collection* sebagai media penyimpanan data dalam pemrograman aplikasi berorientasi obyek  4.12.2 Membuat kode program aplikasi berorientasi obyek yang menerapkan *collection* sebagai media penyimpanan data. | **Nama Siswa :**  **Aurielia Vegayanta** |
| **NIS :7711** |
| **Kelas : XII RA** |

1. **Tujuan** Pembelajaran :
2. Melalui kegiatan mengamati tayangan video tentang konsep dan prosedure *collection* sebagai media penyimpanan data dalam pemrograman aplikasi berorientasi obyek, peserta didik dapat merancang program aplikasi berorientasi obyek dengan penerapan *collection* sebagai media penyimpanan data dengan benar
3. Melalui kegiatan mengamati tayangan video tentang membuat program aplikasi berorientasi obyek yang menerapkan *collection* sebagai media penyimpanan data dalam pemrograman aplikasi berorientasi obyek, peserta didik dapat membuat dan menyajikan program aplikasi berorientasi obyek yang menerapkan *collection* sebagai media penyimpanan data dengan benar.
4. **Alat dan Bahan**
5. Komputer/ Laptop
6. NetBean
7. **Keselamatan Kerja**
8. Pastikan komputer/ Laptop dalam posisi yang baik
9. Pastikan posisi badan aman dari bahaya
10. **Persiapan**
11. Pastikan komputer/ laptop bisa digunakan
12. Pastikan aplikasi netbean yang digunakan suda terinstall di dalam komputer/ Laptop
13. Siapkan buku catatan, untuk mencatat hal yang penting
14. **Materi**

**Java Collections**

**Interface, List, Queue, Sets di Java**

Apa itu collection Java? Collection Java mengacu pada collection objek individu yang direpresentasikan sebagai satu unit. Dengan java collection Anda dapat melakukan semua operasi seperti pencarian, pengurutan, penyisipan, manipulasi, penghapusan, dll., pada collection Java seperti yang Anda lakukan pada data.

Apa itu Framework Collection Java?

Framework collection Java menyediakan arsitektur untuk menyimpan dan memanipulasi sekelompok objek. Framework collection Java mencakup hal berikut:

• Interface

• Class

• Algoritma

**Interface:**

Interface di Java mengacu pada tipe data abstrak. Mereka mengizinkan collection Java untuk dimanipulasi secara independen dari detail representasi mereka. Juga, mereka membentuk hierarki dalam bahasa pemrograman berorientasi objek.

**Class :**

Class di Java adalah implementasi dari Interface collection. Ini pada dasarnya mengacu pada struktur data yang digunakan berulang kali.

**Algoritma:**

Algoritma mengacu pada method yang digunakan untuk melakukan operasi seperti pencarian dan pengurutan, pada objek yang mengimplementasikan Interface collection. Algoritma bersifat polimorfik karena method yang sama dapat digunakan untuk mengambil banyak bentuk atau Anda dapat mengatakan melakukan implementasi yang berbeda dari Interface collection Java.

Mengapa menggunakan collection Java?

Ada beberapa manfaat menggunakan collection Java seperti:

• Mengurangi upaya yang diperlukan untuk menulis kode dengan menyediakan struktur dan algoritme data yang berguna

• Collection Java menyediakan struktur dan algoritme data berkinerja tinggi dan berkualitas tinggi sehingga meningkatkan kecepatan dan kualitas

• API yang tidak terkait dapat melewati Interface collection bolak-balik

• Mengurangi upaya ekstra yang diperlukan untuk mempelajari, menggunakan, dan mendesain API baru

• Mendukung penggunaan kembali struktur dan algoritme data standar

Jadi menurut Anda mengapa membutuhkan collection Java? Framework pengumpulan Java menyediakan pengembang untuk mengakses struktur data yang dikemas serta algoritma untuk memanipulasi data.

**Hirarki Framework Collection Java**

Seperti yang telah kita pelajari, Framework kerja collection Java mencakup Interface dan Class. Sekarang, mari kita lihat hierarki Framework kerja collection Java.



Pada gambar di atas, bagian biru mengacu pada Interface yang berbeda dan bagian kuning mendefinisikan Class. Sekarang, mari kita memahami komponen-komponen ini secara rinci.

**Collection Java: Interface**

**Interface iterator :**

Iterator adalah Interface yang mengulangi elemen. Ini digunakan untuk melintasi daftar dan memodifikasi elemen. Interface Iterator memiliki tiga method yang disebutkan di bawah ini:

1. public boolean hasNext() – Method ini mengembalikan nilai true jika iterator memiliki lebih banyak elemen.
2. objek publik next() – Ini mengembalikan elemen dan memindahkan pointer kursor ke elemen berikutnya.
3. public void remove() – Method ini menghapus elemen terakhir yang dikembalikan oleh iterator.

Ada tiga komponen yang memperluas Interface collection yaitu List, Queue dan Set. Mari kita pelajari tentang mereka secara detail:

* **Collection Java: List**

Daftar adalah Kumpulan elemen yang dipesan yang mungkin berisi duplikat. Ini adalah Interface yang memperluas Interface Collection. Daftar selanjutnya diklasifikasikan menjadi berikut:

1. ArrayList

2. LinkedList

3. Vektor

1. **ArrayList**

ArrayList adalah implementasi dari Interface Daftar di mana elemen dapat ditambahkan atau dihapus secara dinamis dari daftar. Juga, ukuran daftar meningkat secara dinamis jika elemen ditambahkan lebih dari ukuran awal.

**Bentuk Umum penulisan kode program :**

**ArrayList object= new ArrayList();**

Beberapa method dalam array list tercantum di bawah ini:

|  |  |
| --- | --- |
| **Method** | **Deskripsi** |
| boolean add(Collection c) | Menambahkan elemen tertentu ke akhir daftar |
| void add(int index, Object element) | Menyisipkan elemen tertentu pada posisi yang ditentukan. |
| void clear() | Menghapus semua elemen dari daftar ini. |
| int lastIndexOf(Object o) | Mengembalikan indeks dalam daftar kejadian terakhir dari elemen yang ditentukan, atau -1 jika daftar tidak berisi elemen ini. |
| Object clone() | mengembalikan salinan pendek dari ArrayList. |
| Object[] toArray() | Mengembalikan array yang berisi semua elemen dalam daftar. |
| void trimToSize() | Memangkas kapasitas instance ArrayList ini menjadi ukuran daftar saat ini. |

Mari kita memahami ArrayList dengan contoh program:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import java.util.\*;   class ArrayListExample{   public static void main(String args[]){   ArrayList al=new ArrayList();  // creating array list   al.add("Jack");                // adding elements   al.add("Tyler");  al.addAll(Arrays.asList(“Eko”,”Reno”,”Enggar”));   Iterator itr=al.iterator();   while(itr.hasNext()){   System.out.println(itr.next());   }   }   } |

Pada kode di atas, ia akan mengembalikan nama yang telah kita tambahkan menggunakan method add() yaitu:

**Mendongkrak**

**Tyler**

1. **Linked List**

Linked List adalah urutan link yang berisi item. Setiap link berisi koneksi ke link lain.

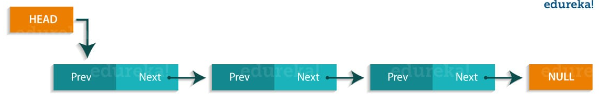
**Bentuk Umum penulisan kode program :**

**objek Linkedlist = new Linkedlist();**

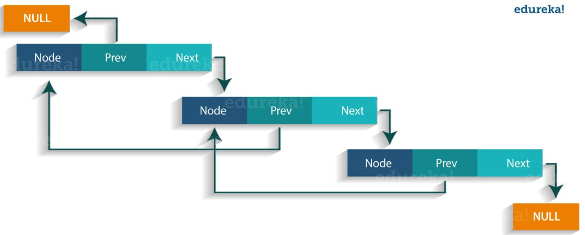
Class Java Linked List menggunakan dua jenis Linked list untuk menyimpan elemen:

* Singly Linked List
* Doubly Linked List

1. **singly Linked list** : Dalam single Linked list, setiap simpul dalam daftar ini menyimpan data simpul dan penunjuk atau referensi ke simpul berikutnya dalam daftar. Lihat gambar di bawah ini untuk mendapatkan pemahaman yang lebih baik tentang single Linked list.



1. **Doubly Linked List**: Dalam Doubly Linked List, ia memiliki dua referensi, satu ke simpul berikutnya dan satu lagi ke simpul sebelumnya. Anda dapat merujuk ke gambar di bawah ini untuk mendapatkan pemahaman yang lebih baik tentang Doubly Linked List.



Beberapa method dalam linked list tercantum di bawah ini:

|  |  |
| --- | --- |
| **Method** | **Description** |
| boolean add( Object o) | Digunakan untuk menambahkan elemen tertentu ke akhir vektor. |
| boolean contains(Object o) | Mengembalikan nilai true jika daftar ini berisi elemen yang ditentukan. |
| void add (int index, Object element) | Menyisipkan elemen pada elemen yang ditentukan dalam vektor. |
| void addFirst(Object o) | Digunakan untuk memasukkan elemen yang diberikan di awal. |
| void addLast(Object o) | Digunakan untuk menambahkan elemen yang diberikan ke akhir. |
| int size() | Digunakan untuk mengembalikan jumlah elemen dalam daftar |
| boolean remove(Object o) | Menghapus kemunculan pertama dari elemen tertentu dari daftar ini. |
| int indexOf(Object element) | Mengembalikan indeks kemunculan pertama dari elemen yang ditentukan dalam daftar ini, atau -1. |
| int lastIndexOf(Object element) | Mengembalikan indeks kemunculan terakhir dari elemen yang ditentukan dalam daftar ini, atau -1. |

Mari kita memahami linked list dengan contoh terprogram :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import java.util.\*;  public class LinkedlistExample{      public static void main(String args[]){          LinkedList<String> al=new LinkedList<String>();// creating linked list          al.add("Rachit"); // adding elements          al.add("Rahul");          al.add("Rajat");          Iterator<String> itr = al.iterator();          while(itr.hasNext()){              System.out.println(itr.next());          }      }  } |

**Output dari program di atas akan menjadi:**

Rachit

Rahul

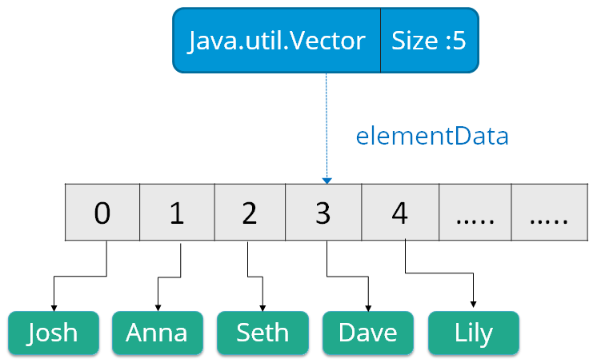
Rajato

**Vektor**

Vektor mirip dengan array, dimana elemen dari objek vektor dapat diakses melalui indeks ke dalam vektor. Vektor mengimplementasikan array dinamis. Juga, vektor tidak terbatas pada ukuran tertentu, ia dapat menyusut atau tumbuh secara otomatis kapan pun diperlukan. Ini mirip dengan ArrayList, tetapi dengan dua perbedaan:

• Vektor disinkronkan.

• Vektor berisi banyak method warisan yang bukan merupakan bagian dari kerangka collection.



**Syntax**:

**Vector object = new Vector(size,increment);**

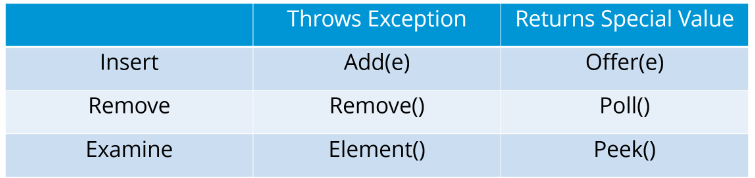
Di bawah ini adalah beberapa method dari Class Vektor:

|  |  |
| --- | --- |
| **Method** | **Description** |
| boolean add(Object o) | Menambahkan elemen yang ditentukan ke akhir daftar. |
| void clear() | Menghapus semua elemen dari daftar ini. |
| void add(int index, Object element) | Menyisipkan elemen tertentu pada posisi yang ditentukan. |
| boolean remove(Object o) | Menghapus kemunculan pertama dari elemen tertentu dari daftar ini. |
| boolean contains(Object element) | Mengembalikan nilai true jika daftar ini berisi elemen yang ditentukan. |
| int indexOfObject (Object element) | Mengembalikan indeks kemunculan pertama dari elemen yang ditentukan dalam daftar, atau -1. |
| int size() | Mengembalikan jumlah elemen dalam daftar ini. |
| int lastIndexOf(Object o) | Mengembalikan indeks kemunculan terakhir dari elemen yang ditentukan dalam daftar, atau -1 jika daftar tidak mengandung elemen apa pun. |

Sekarang, mari kita pindah ke sub tipe berikutnya dari antarmuka Java Collections yaitu Queue.

**Collection Java: Queque (Antrian)**

Queque di Java mengikuti pendekatan FIFO yaitu mengurutkan elemen dengan cara First In First Out. Dalam antrian, elemen pertama dihapus terlebih dahulu dan elemen terakhir dihapus pada akhirnya. Setiap method dasar ada dalam dua bentuk : satu melempar pengecualian jika operasi gagal, yang lain mengembalikan nilai khusus.



Juga, antrian prioritas mengimplementasikan antarmuka Antrian. Elemen-elemen antrian prioritas diurutkan menurut urutan alaminya, atau oleh Pembanding yang disediakan pada waktu konstruksi antrian. Kepala antrian ini adalah elemen terkecil sehubungan dengan pemesanan yang ditentukan.

Di bawah ini adalah beberapa method antarmuka Java Queue:

|  |  |
| --- | --- |
| **Method** | **Description** |
| boolean add(object) | Menyisipkan elemen tertentu ke dalam antrian dan mengembalikan nilai true jika berhasil. |
| boolean offer(object) | Menyisipkan elemen tertentu ke dalam antrian ini. |
| Object remove() | Mengambil dan menghapus kepala antrian. |
| Object poll() | Mengambil dan menghapus kepala antrian, atau mengembalikan null jika antrian kosong. |
| Object element() | Mengambil, tetapi tidak menghapus kepala antrian. |
| Object peek() | Mengambil, tetapi tidak menghapus kepala antrian ini, atau mengembalikan null jika antrian kosong. |

Mari kita pahami antrian prioritas ini dengan contoh terprogram :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | import java.util.\*;  class QueueExample {      public static void main(String args[]){          PriorityQueue<String> queue=new PriorityQueue<String>();          // creating priority queue          queue.add("Amit");          // adding elements          queue.add("Rachit");          queue.add("Rahul");          System.out.println("head:"+queue.element());          System.out.println("head:"+queue.peek());          System.out.println("iterating the queue elements:");          Iterator itr=queue.iterator();          while(itr.hasNext()){              System.out.println(itr.next());          }          queue.remove();          queue.poll();          System.out.println("after removing two elements:");          Iterator<String> itr2=queue.iterator();          while(itr2.hasNext()){              System.out.println(itr2.next());          }      }  } |

Dalam kode di atas, outputnya adalah:

Head : Amit

Head : Amit

iterasi elemen antrian:

amit

Rachit

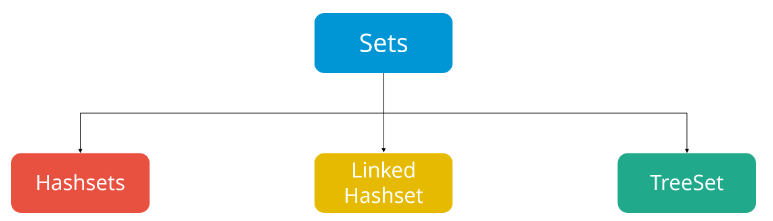
Rahul

setelah menghapus dua elemen:

Rahul

**Collection Java: Set**

Set mengacu pada collection yang tidak dapat berisi elemen duplikat (setiap elemen harus unik). Ini terutama digunakan untuk memodelkan abstraksi himpunan matematika. Set memiliki implementasinya di berbagai Class seperti HashSet, TreeSet dan LinkedHashSet.



Mari kita masuk ke detail masing-masing:

HashSet: Class Java HashSet membuat collection yang menggunakan tabel hash untuk penyimpanan. Hashset hanya berisi elemen unik dan mewarisi Class AbstractSet dan mengimplementasikan antarmuka Set. Juga, ia menggunakan mekanisme hashing untuk menyimpan elemen.

Di bawah ini adalah beberapa method Class Java HashSet:

|  |  |
| --- | --- |
| **Method** | **Description** |
| boolean add(Object o) | Menambahkan elemen yang ditentukan ke set ini jika belum ada. |
| boolean contains(Object o) | Mengembalikan nilai true jika set berisi elemen yang ditentukan. |
| void clear() | Menghapus semua elemen dari set. |
| boolean isEmpty() | Mengembalikan nilai true jika set tidak berisi elemen. |
| boolean remove(Object o) | Hapus elemen yang ditentukan dari set. |
| Object clone() | Mengembalikan salinan dangkal dari instance HashSet: elemen itu sendiri tidak dikloning. |
| Iterator iterator() | Mengembalikan iterator atas elemen dalam set ini. |
| int size() | Mengembalikan jumlah elemen dalam himpunan. |

Mari kita pahami Hashset ini dengan contoh terprogram:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import java.util.\*;  class HashsetExample{   public static void main(String args[]){     HashSet&amp;amp;lt;String&amp;amp;gt; al=new HashSet(); // creating hashSet   al.add("Rachit");                 // adding elements   al.add("Amit");   al.add("jack");   Iterator&amp;amp;lt;String&amp;amp;gt; itr=al.iterator();   while(itr.hasNext()){   System.out.println(itr.next());   }   }   } |

**Output :**

**Amit**

**Rachit**

**Jack**

**Linked Hashset :**

Java LinkedHashSet class adalah tabel Hash dan implementasi daftar Linked dari antarmuka yang ditetapkan. Ini hanya berisi elemen unik seperti HashSet. Linked HashSet juga menyediakan semua operasi set opsional dan mempertahankan urutan penyisipan.

Mari kita pahami Hashset tertaut ini dengan contoh terprogram:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import java.util.\*;  class LinkedHashsetExample{  public static void main(String args[]){  LinkedHashSet&amp;amp;lt;String&amp;amp;gt; al=new LinkedHashSet();  // creating linkedhashset  al.add("Mariana");                            // adding elements  al.add("Rick");  al.add("Sam");  Iterator&amp;amp;lt;String&amp;amp;gt; itr=al.iterator();  while(itr.hasNext()){  System.out.println(itr.next());  }  }  }  } |

Output dari kode di atas akan menjadi:

Mariana

Rick

sama

**TreeSet**

Class TreeSet mengimplementasikan antarmuka Set yang menggunakan pohon untuk penyimpanan. Objek Class ini disimpan dalam urutan menaik. Juga, ia mewarisi Class AbstractSet dan mengimplementasikan antarmuka NavigableSet. Ini hanya berisi elemen unik seperti HashSet. Di Class TreeSet, akses dan waktu pengambilan lebih cepat.

Di bawah ini adalah beberapa method Class Java TreeSet:

|  |  |
| --- | --- |
| **Method** | **Description** |
| boolean addAll(Collection c) | Tambahkan semua elemen dalam collection yang ditentukan ke set ini. |
| boolean contains(Object o) | Mengembalikan nilai true jika set berisi elemen yang ditentukan. |
| boolean isEmpty() | Mengembalikan nilai true jika set ini tidak berisi elemen. |
| boolean remove(Object o) | Hapus elemen yang ditentukan dari set. |
| void add(Object o) | Tambahkan elemen yang ditentukan ke set. |
| void clear() | Tambahkan semua elemen dalam collection yang ditentukan ke set ini. |
| Object clone() | Mengembalikan nilai true jika set berisi elemen yang ditentukan. |
| Object first() | Mengembalikan nilai true jika set ini tidak berisi elemen. |
| Object last() | Hapus elemen yang ditentukan dari set. |
| int size() | Tambahkan elemen yang ditentukan ke set. |

Mari kita pahami Hashset tertaut ini dengan contoh terprogram:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import java.util.\*;  class TreeSetExample{  public static void main(String args[]){  TreeSet&amp;amp;lt;String&amp;amp;gt; al=new TreeSet&amp;amp;lt;String&amp;amp;gt;();  // creating treeSet  al.add("John");                            // adding elements  al.add("Sam");  al.add("Rick");  Iterator&amp;amp;lt;String&amp;amp;gt; itr=al.iterator();  while(itr.hasNext()){  System.out.println(itr.next());  }  }  } |

Output :

John

Rick

Sama

**Rangkuman :**

HashSet menyimpan elemen dalam urutan acak.

LinkedHashSet menyimpan elemen menurut urutan penyisipan.

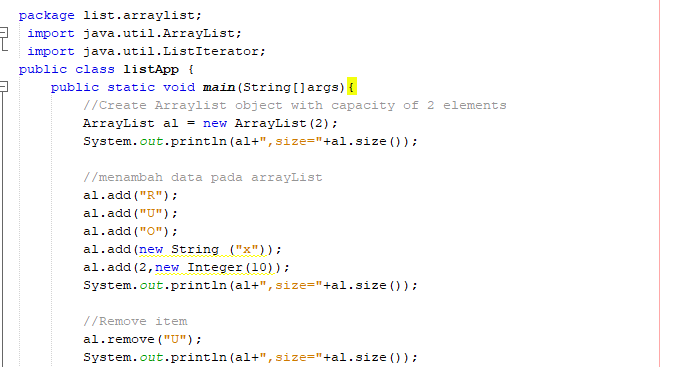
TreeHashSet menyimpan menurut urutan alami.

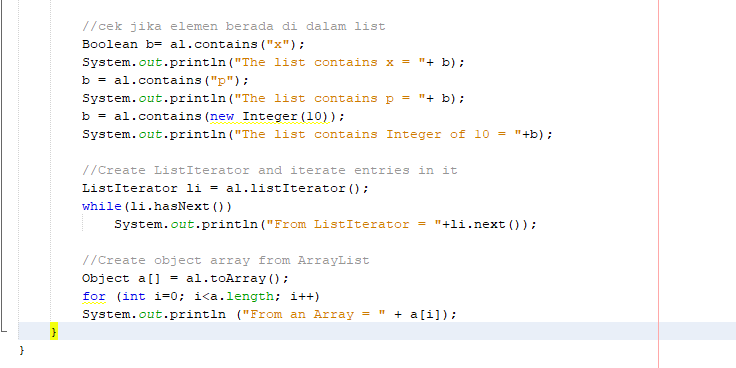
Implementasi dari ArrayList dan Vector ini mirip seperti array, dimana kita dapat menggakses anggotanya melalui indeksnya. Kelebihannya adalah ia dapat menyesuaikan ukurannya sesuai dengan kebutuhan. Perbedaan dari kedua kelas ini, vector secara internal telah mendukung sinkronisasi, sedangkan ArrayList secara internal tidak mendukung sinkronisasi.

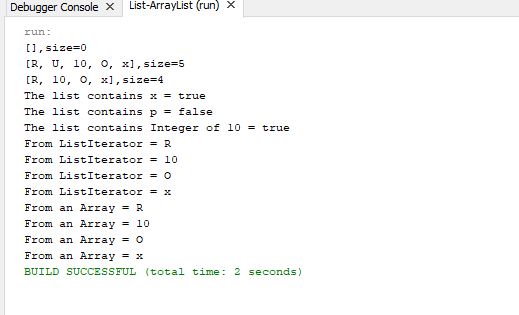
**Latihan ArrayList :**

1. Buat projek baru pada NetBeans.

* Pilih **File**->**New Project (Ctrl+Shift+N)**.
* Window **New Project** akan muncul.
* Pada bagian **Choose Project** window, pilih **Java**pada bagian **Categories**dan **Java Application** pada bagian **Projects**.
* Klik **Next**.
* Pada bagian **Name and Location** window, untuk teksfield **Project Name** ketik **List-ArrayList**sebagai nama projek.
* Untuk teksfield **Create Main Class** gunakan seperti yang muncul pada IDE.
* Klik **Finish**.

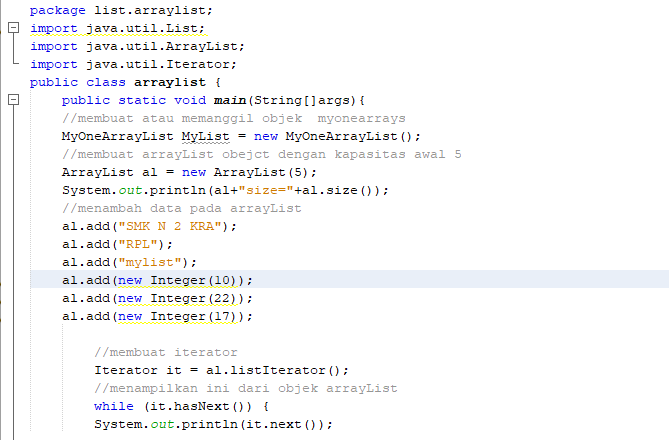
****

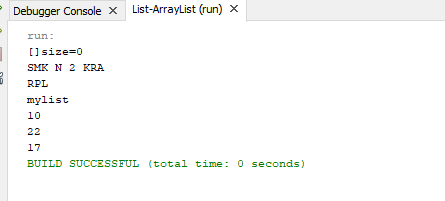
****

****

1. Sebagai bahan latihan, lakukan hal di bawah:
   * + - Buat projek NetBeans dengan nama **MyNamaSiswa\_ArrayList**

* Buat objek **ArrayList**dengan kapasitas awal 5
* Tambahkan hal-hal di bawah ke dalam objek **ArrayList**.
  + 2 String objek
  + 1 MyOneArrayList objek (Anda buat MyOneArrayList.java terlebih dahulu)
  + 3 Integer objek
* Buat objek Iterator dari objek **ArrayList**dan iterate isi dari ArrayList tersebut
* Kemudian tampilkan isi dari objek **ArrayList**tersebut

****

****

\*\*SELAMAT BERUSAHA\*\*

Kamu BISA