**Catatan Pembelajaran Java**

**Penggunaan Library Java**

|  |  |
| --- | --- |
| Menggunakan seluruh library pada “util” | import java.util.\*; |
|  |  |
| Mendapatkan input dari terminal | import java.util.Scanner; |
| Penggunaan Array | Import java.util.Arrays; |
| Penggunaan String | import java.lang.String; |
| String Builder | Import java.lang.StringBuilder; |
| Mendapatkan input dari file | Import java.io.FileInputStream; |
|  |  |

**Mengambil input / user input**

1. Panggil library scanner/ untuk input
2. Buat **Objek** scanner yang terhubung dengan kelas dari scanner

*Scanner vaiabel = new Scanner(System,in);*

1. Buat variabel yang akan menyimpan input dari user

*Variabel\_Input = variabel.next();* => next() digunakan untuk tipe data string,

nextInt() digunakan untuk tipe data Integer,

untuk tipe data lain juga berbeda.

**Arrays**

Pemanggilan untuk menggunakan fungsi arrays :

java.util.Arrays; => libary array di Java

Sifat array :

1. Jika kita mendeklarasikan array baru dan mengisi nilainya sama dengan array yang lama maka yang dilakukan adalah **menempatkan array baru pada alamat yang sama** dengan array lama. Contoh :

Int[] array1= {1,2,3,4,5};

Int[] array2 = new int[5];

array2 = array1; //dengan begini isi array2 sama dngan array1, yang mana alamat

array 2 disamakan dengan alamat memori array1

tetapi jika kita **mengubah salah satu isi** di array2 maka array1 **juga ikut berubah**.

Hal ini terjadi karena isi array yang diubah pada array2 merupakan array yang sama dengan isi array pada array 1 karena **alamat yang digunakan sama**.

1. Untuk mengambil isi dari array lama dapat menggunakan metode **for loop** sehingga array baru **tidak di tempatkan di alamat memori yang sama** dengan array yang lama.

Selain *for loop* dapat juga menggunakan perintah aray ***Arrays.copyOf(nama array)*.**

|  |  |
| --- | --- |
| Arrays.toString(*nama array*); | Mengubah array menjadi string, betuk string :  [1,2,3,4,5] |
| Arrays.deepToString(*nama array*); | Mengubah array 2 dimensi menjadi sebuah string, bentuk string: [[1,2], [3, 4]] |
| Int[] arrayB = Arrays.copyOf(*arrayA*, 5)  “5 => panjang isi array yang ingin di copy” | Membuat isi arrayB menjadi sama dengan isi pada arrayA, tetapi yang disalin adalah isi nya bukan menyamakan alamatnya. Sehingga jika arrayB berubah maka arrayA tidak ikut berubah, begitupun sebaliknya. |
| arrayB = Arrays.copyOfRange(arrayA, 2, 5)  “2, 5 => array yang di copy merupakan array pada index ke-2 sampe ke-5” | Membuat isi array b menjadi sama dengan isi pada arrayA menggunakan range tertentu |
| Arrays.fill(arrayB, 2);  “2 => angka yang akan diisikan ke dalam array” | Mengisi semua tempat apada array dengan angka yang ditentukan |
| Arrays.equals(arrayA,arrayB) | Melakukan pengecekkan isi dari array. Akan mengembalikan nilai *true* atau *false* |
| Arrays.compare(araryA, arrayB) | Membandingkan isi dari kedua arary. Mengembalikan nilai 1 jika array pada parameter pertama yang lebih besar, nilai -1 jika parameter kedua yang lebih besar, dan mengembalikan 0 jika nilainya sama. |
| Arrays.mismatch(arrayA, arrayB) | Melihat index dari isi array yang berbeda. |
| Arrays.sort(arrayA) | Melakukan sorting / pengurutan isi array dari nilai terendah ke nilai yang tertinggi. |
| arayA.length | Menghitung panjang array |

**String**

Library => *import java.lang.String;*

Sifat string :

1. Ketika variabel string baru yang isi nya disamakan dengan string lama kemudian isinya diubah maka string tersebut akan menggunakan alamat **memori yang baru**.

( **Hal ini berbeda dengan sifat Array ).**

1. Variabel yang diisi dengan **string saat dideklarasikan** akan masuk ke dalam **alamat memori literal**. Jika variabel string yang isinya sama dilakukan pengecekkan menggunakan IF maka hasilnya akan **true**.

*If( stringA == stringB)* => akan menghasilkan nilai True.

1. Tetapi jika varabel diisi dengan menggunakan string yang **diinputkan oleh user** ketika program sudah berjalan, variabel tersebut akan diletakkan di alamat **memori** **heap.**

Artinya memori yang digunakan **sama** dengan memori untuk menyimpan array, yang mana jika dilakukan pengecekan isi yang sama menggunakan **IF** maka menghasilkan nilai **false.** Untuk melakukan pengecekannya dapat menggunakan perintah **equals**, bisa dilihat pada tabel di bawah.

|  |  |
| --- | --- |
| *Nama\_string*.charAt(5);  “5 => mengambil index ke 5” | Mengambil karakter dari string pada index tertentu |
| *Nama\_string*.substring(10, 15);  “10, 15 => range index yang diambil 10-15” | Mengambil karakter dari string pada index range index tertentu lalu mengembalikan dalam bentuk string. |
| *namaString*.toUpperCase(); | Mengubah isi string menjadi hurup kapital semua / Uppercase |
| *namaString*.toLowerCase(); | Mengubah isi string menjadi hurup kecil semua / Lowercase |
| *namaString*.equals(*namaString2)*; | Untuk mengecek apakah isi string pertama sama dengan string kedua |
| *namaString*.compareTo(*namaString2);* | Untuk mengetahui perbandingan isi string menurut abjad dan pengecekan dilakukan dari karakter pertama yang **berbeda**. Fungsi ini akan mengembalikan nilai integer. Nilai ini adalah jarak antara karakter **String1 dari String2**. Misal **string1 = Alam** dan **string2 = Cerah**, maka fungsi akan mengembalikan nilai **-2.** Jika variabel yang dicek dibalikan  **String2 compare string1** , maka akan mengembalikan nilai **2.** |

**String Builder**

String builder adalah sebuah library pada Java untuk penggunaan string dengan pengalokasian memori yang lebih ringan. Dengan string builder dapat membuat penggunaan string yang dinamis / berubah-ubah menjadi lebih mudah.

Cara penggunaan string builder :

1. Panggil library string builder
2. Buat **Objek** yang menampung string builder

StringBuilder *objekString* = new StringBuilder(“Halo”); => isi objek “halo”

1. Lalu isi objek dapat diubah tanpa mengganti alamat memori objek.

|  |  |
| --- | --- |
| *objekString*.append(“ kata baru”); | Menambahkan isi string baru ke dalam objek. String ditambahkan di belakang isi string sebelumnya. |
| *objekString*.insert(10, “tambah”);  “10 => letak index” | Menambahkan isi string baru pada index tertentu dari isi string sebelumnya. |
| *objekString*.delete(10, 22);  “10, 22 => range letak index” | Menghapus isi string pada range index tertentu |
| *objekString*.setCharAt(10, “S”);  “10 => letak index” | Mengganti sebuah karakter pada range tertentu. |
| *objekString*.replace(10, 18, “ ganti” );  “10,18 => letak index”  “ganti => karakter baru untuk pengganti” | Mengganti banyak karakter sesuai range tertenru menjadi karakter baru yang diberikan. |
| String *variabel* = *objekString*.toString() | Mengubah **String Builder** menjadi **String** biasa |

**Mengambil input dari file**

1. Panggil library input file
2. Buat objek baru dengan kode :

FileInputStream *variabel* = new FileInputStream(“*nama file yang dituju.txt”)*;

1. Dalam menggunakan file input, wajib menyertakan **Exception**.

Contoh :

try {

FileInputStream *variabel* = new FileInputStream(“*nama file yang dituju.txt”)*;

} catch (Exception e){

System.err.println(e);

}

1. Lokasi file yang akan menjadi input harus selevel dengan letak direktori program.
2. Untuk membaca input dilakukan dengan :

*variabel*.read();

1. Input dibaca per byte/ 8 bit, atau jika diubah menjadi karakter dengan menambahkan (*char)*. Menjadi : (*char)variabel*.read();
2. Jika ingin membaca semua item yang ada pada file, maka perlu menggunakan perulangan untuk mencetak setiap karakter yang ada pada file.
3. *variabel*.read(); jika dilakukan secara berulang akan menghasilkan karakter secara berurutan dari karakter awal sampai seterusnya.
4. Contoh jika menggunakan perulangan **While** :

Int buffer = *variabel.*read(); **// deklarasikan dulu variabel penampung nilai input awal**

while( buffer != -1) ]{ **// -1 adalah angka pada Byte Stream yang berarti kosong**

System.out.print(buffer);

buffer = *variabel*.read();

}

**\*Catatan : Selalu ingat untuk melakukan colse ketika sudah selesai menggunakan file input :**

***variabel.*close();**

Selain menggunakan kode di atas, untuk close input stream dapat dilakukan pada bagian try exception, seperti ini :

**//deklarasikan objek input di dalam kurung try**

try( FileInputStream *variabel* = new FileInputStream(”nama file.txt”)) {

isi tidakan dalam try atau penggunaan file input

}

**//Dengan kodingan ini maka setelah bagian pada try selesai dilakukan maka file input akan otomatis di close.**

**Jika ingin di gabungkan dengan menutup file output dapat dilakukan dengan menambahkan titik koma ( ; ) lalu tambahkan objek output. Seperti kode dibawah**

try( FileInputStream *variabel* = new FileInputStream(”nama file.txt”);

FileOutputStream *variabel* = new FileOutputStream(”nama file.txt”)) {

isi tidakan dalam try atau penggunaan file input

}

**Membuat file Output**

1. membuat file output dapat ditulis mirip dengan mengambil file input

pertama buat objek output:

*variabel* = new FileOutputStream(”*Nama file output.txt”*);

1. jika objek file output dijalankan dan tidak terdapat file output di direktori yang sama dengan program, maka saat program dijalakan akan membuat file outpu secara otomatis. Mesikupun nilai di file ouput kosong.
2. Untuk memasukan data pada file output dapat dilakukan seperti ini :

*variabel.*write(*isi);*

**Mengambil input dan membuat ouput dari File dengan Character Stream**

Berbeda dengan cara diatas yang mengambil input menggunakan Byte stream, yang mana input yang dianbil per 8 bit sedangkan dengan cara ini mengambil input berupa karakter.

Character stream digunakan untuk file input yang mana isi di dalam file terdapat karakter yang tidak ada di dalam karakter ASCII code, contoh tulisan Jepang, tulisan cina, Arab, dll.

Cara menggunakan Character Stream :

1. Pembuatan objek :

FileReader *variabel* = new FileReader(“*nama file.txt*”);

1. Untuk menampilkan data inputnya sama caranya dengan menampilkan input dari Byte Stream.
2. Untuk **output** pembuatan objek nya dilakukan dengan seperti ini :

FileWriter *variabel*  = new FileWriter(“*Nama file.txt”*);