STRING

OBJEKTIF :

1. Mahasiswa Mampu Memahami String pada Java.
2. Mahasiswa Mampu Menggunakan *Software* IntelliJ IDEA untuk Membuat Program dengan Operasi String.

PENDAHULUAN

String adalah rangkaian simbol alphabet, angka, spasi, dan karakter lainnya. Berikut contoh mendeklarasikan *string* :

```
String nama = “Ani Lestari”;
```

10.1 DESKRIPSI STRING

Di Java, terdapat dua kelas yang memodelkan *string* yaitu :

1. *String*, untuk *string* konstan yang tidak berubah isinya setelah penciptaannya.
2. *StringBuffer*, untuk string yang memerlukan banyak manipulasi.

10.2 IMMUTABLE STRING

Kelas *String* bersifat *immutable*, yaitu ketika *String* dibuat maka nilai yang ada pada *String* tidak dapat diubah (konstan) secara langsung.

10.3 CONSTRUCTOR STRING

Class String mempunyai 15 *constructor* sehingga terdapat beragam cara eksplisit menciptakan objek *string*. Contoh *Constructor* untuk kelas *String* sebagai berikut :

- *String()*

Constructor ini dapat digunakan untuk menciptakan string baru bernilai kosong (null).

- *String (String value)*

Constructor ini dapat digunakan untuk menciptakan *string* baru dan memiliki nilai.

- *String (char value [])*

Constructor ini berupa *String* dari *array* karakter.

- *String (char value [], int offset, int count)*

Constructor ini berupa *String* dari sub *array* karakter dan memiliki parameter untuk pengambilan karakter.

- *String (byte ascii[], int hibyte, int offset, int count)*

Unicode string dari sub *array byte* menggunakan *hibyte* sebagai *high byte* untuk tiap karakter Unicode.

- *String (byte ascii[], int byte)*

Unicode string dari *array byte* menggunakan *hibyte* sebagai *high byte* untuk tiap karakter *Unicode*.

- *String (StringBuffer buffer)*

String dari objek *stringBuffer*

10.4 OPERASI STRING

Setelah menciptakan objek baru pada *String*, maka kelas *String* ini dapat melakukan metode-metode yang dapat digunakan untuk memanipulasi *string*. Beberapa metode yang bermanfaat di kelas String sebagai berikut :

1. *length()* : *method* ini digunakan untuk menghitung jumlah karakter pada variabel.
2. *charAt()* : *method* ini mengembalikan karakter yang terletak di indeks *String* yang ditentukan. Indeks *string* mulai dari nol.
3. *startsWith ()* : *method* digunakan untuk memeriksa kondisi awalan pada nilai atau karakter dari tipe data *String.*
4. *endsWith()* : *method* ini digunakan untuk memeriksa kondisi akhiran pada nilai atau karakter dari tipe data *String*. Kedua *method* tersebut mengembalikan nilai *boolean*, berupa *true* atau *false*.
5. *indexOf()* : *method* ini digunakan untuk menemukan kemunculan pertama karakter tertentu pada *String* dengan memberikan nilai *index*.
6. *lastIndexOf()* : *method* ini digunakan untuk mengembalikan nilai *index* kemunculan terakhir suatu karakter dalam *String* tertentu.
7. *substring ()* : *method* ini digunakan untuk mengambil sebagian atau beberapa potong karakter dari isi *String*.
8. *equals ()* : *method* tersebut digunakan untuk membandingkan 2 buah variabel yang bertipe data *String*.
9. *equalsIgnoreCase ()* : *method* ini serupa dengan *equals*(), namun bersifat *case insensitivity.*
10. *compareTo()* : *method* ini memungkinkan untuk membandingkan *object* bilangan yang dipanggil melalui *method* untuk argumen.
11. *concat()* : *method* ini digunakan untuk menggabungkan *string*.
12. *trim()* : *method* tersebut berfungsi untuk menghapus spasi dari bagian awal dan juga akhir.
13. *toLowerCase()* dan *toUpperCase()* : *method* ini digunakan untuk mengonversi semua karakter pada objek *String* menjadi *lowercase* dan *uppercase*.
14. *valueOf()* : *method* ini berfungsi untuk mengonversi nilai dari tipe data menyerupai *int, boolean* dan lain sebagainya, atau tipe data objek lain ke dalam tipe data *String*.

Contoh sintaks penggunaan operasi *String* :

- *int length ()*
- *char charAt (int index)*
- *boolean startsWith(String prefix)*
- *boolean startsWith(String prefix, int toffset)*
- *boolean endsWith(String suffix)*
- *int indexof (int ch)*
- *int indexof (int ch, int fromIndex)*
- *int indexOf (String str)*

- *int indexof (String str, int fromIndex)*
- *int lastIndexof (int ch)*
- *int lastIndexof (int ch, int from Index)*
- *int lastIndexof (String str)*
- *int lastIndexof (String str, int fromIndex)*
- *String substring (int beginIndex)*
- *String substring (int beginIndex, int boolean equals (Object anobject)*
- *boolean equalsIgnoreCase(String ring)*
- *int compareTo (String anotherString)*
- *String concat(String str)*
- *String replace (char oldChar, char newChar)*
- *String trim ()*
- *String toLowerCase ()*
- *String toUpperCase()*

10.5 METHOD LENGTH(), SUBSTRING(), CONCAT(), DAN REPLACE()

10.5.1 LENGTH()

Method length() digunakan untuk memperoleh panjang dari *string*. Berikut contoh penggunaan *method length*() :

```
String aString = "Integrated Laboratory";
int len = aString.length();
```

Variabel len akan bernilai 21, yaitu panjang nilai a*String* (termasuk spasi).

10.5.2 *SUBSTRING*()

Substring digunakan untuk mengambil isi sebagian dari variable *String* atau mengambil potongan beberapa karakter dari sebuah isi *String*. Dimana, pada parameter pertama merupakan *index* awal dari *String* dan parameter kedua merupakan batasan *index* yang akan diambil. Sintaks penggunaan substring :

```
String substring(index_awal,index_akhir);
```

Contoh :

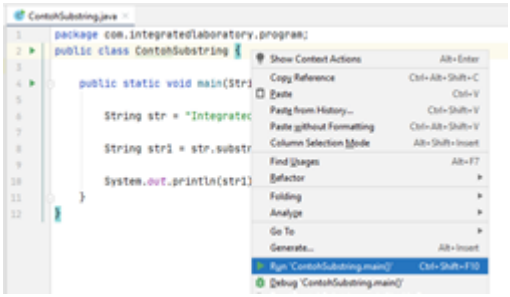
```
package com.integratedlaboratory.program;
public class ContohSubstring {
      public static void main(String[] args) {
       String str = "Integrated Laboratory";

       String str1 = str.substring(0,7);

       System.out.println(str1);
     }
}
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :

Pada contoh di atas, Index 7 adalah karakter ‘t’ tetapi yang terambil karakter ‘a’ karena pengambilan index akhir pada substring bersifat exclusive. Sehingga output program di atas adalah "Integra".

10.5.3 *CONCAT*()

Java menggunakan tanda ‘+’ untuk menyambungkan dua *string*. Apabila saat menyambungkan *string* dengan nilai yang bukan bertipe *string* maka nilai tersebut dikonversi terlebih dahulu menjadi *string*. Setiap objek Java dapat dikonversi menjadi *string*. Sintaks penggunaan concat :

```
String concat(String str);
```

Metode *concat*() digunakan untuk menyambung dua objek *String*. *String* yang dispesifikasikan di parameter *str* kemudian disambu

ng ke akhir objek *string*.

Contoh:

```
String s1 = new String ("Ani sedang membaca");
String s2 = new String (s1 + "koran");
String s3 = s1.concat ("koran");
```

Objek *string* lebih dulu diciptakan dengan nilai "Ani sedang membaca ".

- Contoh penyambungan pertama menunjukkan bagaimana dua *String* dapat disambung menggunakan operator '+'.
- Contoh kedua menunjukkan bagaimana dua *String* dapat disambung menggunakan metode *concat*().
- Pada kedua contoh, *string* yang dihasilkan adalah "Ani sedang membaca koran".

10.5.4 *REPLACE*()

Metode *replace* () mengganti karakter. Semua kemunculan *oldChar* diganti *newChar*.

```
String replace (char oldChar, char newChar);
```

Contoh:

```
 String s4 = s3.replace ('k', 'm');
```

Hasilnya adalah *string* " Ani sedang membaca moran ".

REFERENSI

[1] Hariyanto, Bambang. 2010. Esensi-Esensi Bahasa Pemrograman Java Revisi Ketiga. Bandung: Informatika Bandung.

[2] Tutorialspoint. Java-Strings. Diambil dari : https://www.tutorialspoint.com/java/java_strings.htm (24 Juli 2020)