Bab 6

String Manipulation

Tujuan

1. Memahami Konsep *String* pada Java
2. Memahami *Reference Equality* dan *Value Equality* pada *String*
3. Mampu memanipulasi *String*
4. Mampu memanipulasi *String* dengan *Method* pada *Class String*

Pada bab Tipe Data, telah dibahas tentang tipe data *reference*. *String* merupakan salah satu tipe data *reference* dalam java yang berarti string juga dianggap sebagai objek. *String* juga sering dikenal sebagai *Array of Char* atau kumpulan karakter.

6.1 Deklarasi dan Inisialisasi String

Deklarasi dan inisialisasi tipe data *String* dapat dilakukan dengan cara seperti tipe data primitif yang nilainya ditandai dengan **“”** (Kutip Dua) atau dideklarasikan seperti cara menginstansikan objek.

```
String prodi = “Ilmu Komputer”;
String prodi = new String(“Ilmu Komputer”);
```

Drawing 29: Deklarasi dan Inisialisasi String dengan 2 Cara

Cara deklarasi dan inisialisasi *String* yang pertama dikenal sebagai *String Literal* dimana *string* yang dibuat dengan cara ini akan mengambil nilai *string* dari *string pool* jika sudah ada dan menyimpannya ke *string pool* jika belum ada. Cara kedua dikenal sebagai *String Object*, dimana setiap kali *string* dibuat, berarti akan membuat objek baru dan memiliki alamat memori yang berbeda.

6.2 String dan Array of Char

Di dalam *class String*, terdapat method *toCharArray()* yang akan membuat *array* bertipe data *char* dengan susunan dan panjang yang sama dari sebuah *string*, sebaliknya, sebuah char *array* dapat dijadikan objek *string*.

```
String prodi = “Ilmu Komputer”;
Char[] prodiChar = prodi.toCharArray();
String prodiString = new String(prodiChar);
```

Drawing 30: Membuat Array Char dari String dan String dari Array Char

Selain diubah ke dalam *array of char*, *string* juga dapat dimanipulasi layaknya *array* menggunakan *method charAt(i)* dimana ***i*** adalah indeks yang mulai dari nol, *prodi.charAt(6)* pada kode di atas akan dikembalikan sebuah *char* dengan nilai *‘o’*, perlu diingat bahwa spasi juga terhitung sebagai satu char.

6.3 String Format

Dengan menggunakan *method format()* atau *printf()*, *string* dapat dimanipulasi formatnya sesuai kebutuhan seperti mengatur banyaknya angka di belakang koma, banyaknya spasi, atau menampilkan format *biner*, *octal*, atau *hexa* dari sebuah bilangan desimal.

```
String.format(“%format, %format”, value, value, ...);
System.out.printf(“%format, %format”, value, value, ...);
```

%format merupakan *format specifiers* atau penentu format yang diinginkan, sedangkan ***value*** adalah data yang ingin diformat, jumlah dan urutan *format specifier* harus sama dengan jumlah dan urutan *value*.

```
int n = 90;
String oct = String.format(“%o”, n);
String hex = String.format(“%x”, n);
System.out.printf(“Desimal\t: %d\nOctal\t: %s\n”, n, oct);
System.out.printf(“Hexa\t: %s\n”, hex);
```

Drawing 31: Menampilkan Angka 90 dalam Format Octal dan Hexa

Berikut beberapa *format specifier* dalam java:

Format Specifier	Tipe Data	Fungsi
%d	Integer	Menampilkan Angka
%o	Integer	Menampilkan Angka Octal
%x	Integer	Menampilkan Angka Hexa
%s	String	Menampilkan String
%n		Menampilkan new line
%f	Floating Point	Menampilkan Angka Pecahan
%c	Char	Menampilkan Unicode Character
%a	Floating Point	Manampilkan Angka Pecahan Hexa

Table 8: Beberapa Format Specifier yang Bisa Digunakan

6.4 Reference Equality dan Value Equality pada String

Untuk tipe data primitif, perbandingan nilai dapat diketahui menggunakan operator "==", sedangkan pada *string* terdapat sedikit perbedaan, jika *string* yang dibandingkan dibuat dengan *String Literal*, maka dapat menggunakan operator "==", karena operator ini hanya akan membandingkan alamat memori sebuah objek (*Reference Equality*), sedangkan jika string dibuat menggunakan *String Object*, maka dibutuhkan *method equals()* untuk membandingkannya, karena *method* ini akan membandingkan konten dari *string* (*Value Equality*).

```
String a = "ilkom";
String b = new String("ilkom");
String c = "ilkom";
System.out.println(a == b);
System.out.println(a.equals(b));
System.out.println(c == a);
System.out.println(c == b);
```

Drawing 32: 3 Nilai String yang Sama dengan hasil Perbandingan yang Berbeda

6.5 Method pada Class String

Berikut beberapa *method* pada *class String*, untuk lebih lengkapnya dapat dilihat di : *https://docs.oracle.com/javase/7/docs/api/java/lang/String.html*

Method	Fungsi
charAt(*i*)	Mengembalikan *char* pada indeks ke *i*
equals(*s*)	Mengembalikan *boolean* hasil perbandingan dengan objek *s*
equalsIgnoreCase(s)	Mengembalikan *boolean* hasil perbandingan dengan objek *s* dengan mengabaikan *case sensitive*
isEmpty()	Mengecek apakah sebuah *string* itu kosong
length()	Mengembalikan *int* dari panjang *String*
trim(s)	Menghapus semua karakter *s* dalam *String*
split(s)	Membagi *string* menjadi *array string* berdasarkan karakter *s*
toUpperCase()	Mengubah *string* menjadi huruf kapital
toLoweCase()	Mengubah *string* menjadi huruf kecil
valueOf(s)	Mengubah objek *s* menjadi *String*

Table 9: Beberapa Method Class String