# STRING AS A Class

**Sebuah teks-file pengingat tentang string.**

**Memakai bahasa pemrograman java.**

**Pengantar Untuk mengenal definisi String**

**Fachri .R**

**18, Februari 2023**

**Bandung, indonesia.**

# Definisi String

**A. Apa itu string?**

Dalam pemrograman, string adalah tipe data yang digunakan untuk merepresentasikan teks atau urutan karakter. string terdiri dari satu atau lebih karakter, seperti huruf, angka, simbol, dan spasi. Contoh syntax string “Hello World!”.

String biasanya digunakan untuk merepresentasikan data teks seperti nama, alamat, pesan, atau kode sumber. String juga sering digunakan dalam manipulasi data dan operasi file dalam pemrograman.

Dalam beberapa bahasa pemrograman, string dinyatakan dalam tanda kutip tunggal (‘...') atau tanda kutip ganda ("..."), tergantung pada konvensi yang digunakan oleh bahasa pemrograman tersebut. Misalnya dalam bahasa pemrograman java, string di- Implementasikan dengan tanda kutip dua (“ “). sedangkan dalam bahasa pemrograman python, string dapat di-Implementasikan dengan tanda kutip dua(“ ”), dan tanda kutip satu(‘ ‘).

namun string dalam java dinyatakan sebagai Kelas(class) bawaan. berarti bahwa setiap kali kita membuat string, kita membuat objek dari kelas String. Konsep dasar dalam pemrograman berorientasi objek (OOP) adalah bahwa data dan fungsi terkait terkumpul dalam objek yang dapat digunakan untuk melakukan tugas tertentu. Dengan mendefinisikan string sebagai kelas, Java menyediakan berbagai metode yang dapat digunakan untuk memanipulasi string, seperti mencari karakter tertentu, memotong bagian string, atau mengubah ukuran huruf.

**B. Apa itu String Pool?**

String pool adalah area di memori JVM (Java Virtual Machine) yang menyimpan string yang telah dibuat oleh program. Ketika program membuat string dengan literal, seperti "Hello" atau "World", JVM akan mengecek apakah string tersebut sudah ada di string pool. Jika iya, JVM akan menggunakan string yang sudah ada di pool. Jika tidak, JVM akan menambahkan string tersebut ke pool.

Tujuan dari string pool adalah untuk mengoptimalkan penggunaan memori pada program. Karena string adalah objek yang sering digunakan dalam program, dan karena string yang sama mungkin dibuat berulang kali dalam program, maka dengan menggunakan string pool, program dapat menghindari penggunaan memori yang berlebihan.

String pool juga memiliki pengaruh pada cara perbandingan string. Ketika kita membandingkan dua string menggunakan operator ==, Java akan membandingkan alamat memori dari dua objek string tersebut. Jika keduanya terdapat pada objek yang sama di string pool, maka perbandingan akan menghasilkan nilai true. Namun, jika keduanya mengacu pada objek yang berbeda, meskipun string-nya sama, maka perbandingan akan menghasilkan nilai false.

Penting untuk diingat bahwa string pool tidak selalu digunakan. String yang dibuat menggunakan konstruktor new String(), misalnya, tidak akan disimpan di string pool, melainkan di area memori biasa. Oleh karena itu, perlu hati-hati dalam penggunaan string di program, dan perlu mempertimbangkan efek penggunaan string pool terhadap kinerja dan memori program.

**C. Struktur dan Format String**

Format String memiliki tiga spesifikasi: %s untuk string, %d untuk bilangan bulat, dan %f untuk bilangan desimal. Argumen tambahan setelah String format akan dimasukkan ke dalam spesifikasi yang sesuai. format String dapat dilakukan menggunakan kelas String.format() atau printf().

Perbedaan antara String.format() dan printf() adalah bahwa printf() akan langsung mencetak String format ke output tanpa perlu menampungnya terlebih dahulu di dalam variabel.

*Dokumentasi Lengkap tentang Format String ada di oracle: https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html*

Beberapa operasi dalam string:

- **Concatenation** (Penggabungan):

Menggabungkan dua atau lebih string menggunakan operator *"+"* atau metode *concat()*.

Contoh:

*String str1 = "Hello";*

*String str2 = "World";*

*String result = str1 + " " + str2; // Output: "Hello World"*

*String result2 = str1.concat(" ").concat(str2); // Output: "Hello World"*

*-* Length: Mengembalikan jumlah karakter dalam string menggunakan metode length().

Contoh:

*String str = "Hello World";*

*Int len = str.length(); // Output: 11*

*-* Substring: Mengambil bagian tertentu dari string menggunakan metode substring() dengan parameter indeks awal dan akhir.

Contoh:

String str = "Hello World";

String result = str.substring(6, 11); // Output: "World"

- Replace: Mengganti karakter atau substring tertentu dalam string dengan karakter atau substring lain menggunakan metode replace().

Contoh:

String str = "Hello World";

String result = str.replace("World", "Universe"); // Output: "Hello Universe"

- Split: Memisahkan string menjadi array substring menggunakan metode split() dengan separator tertentu.

Contoh:

String str = "apple,banana,grape";

String[] arr = str.split(","); // Output: {"apple", "banana", "grape"}

- Trim: Menghilangkan karakter spasi pada awal dan akhir string menggunakan metode trim().

Contoh:

String str = " Hello World ";

String result = str.trim(); // Output: "Hello World"

- Format: Membuat string format yang diformat secara khusus dengan metode String.format().

Contoh:

String str = "John";

int age = 30;

String result = String.format("My name is %s and I'm %d years old.", str, age);

// Output: "My name is John and I'm 30 years old."

- Case Conversion: Mengubah case (upper case atau lower case) dari string menggunakan metode toUpperCase() atau toLowerCase().

Contoh:

String str = "Hello World";

String result1 = str.toUpperCase(); // Output: "HELLO WORLD"

String result2 = str.toLowerCase(); // Output: "hello world"

Operasi lainnya pada string di Java juga termasuk mencari karakter atau substring tertentu dalam string menggunakan metode indexOf() atau lastIndexOf(), mengonversi s tring menjadi array karakter menggunakan metode toCharArray(), dll.

**D. String Builder**

StringBuilder sangat berguna dalam situasi di mana Anda perlu melakukan manipulasi pada string secara dinamis dan perlu menghindari overhead yang terkait dengan pembuatan string baru setiap kali operasi dilakukan. Namun, jika Anda tidak memerlukan mutabilitas string, maka String yang immutable (tidak dapat berubah) tetap merupakan pilihan yang lebih baik.

Anda dapat mengubah StringBuilder ke dalam String dengan menggunakan metode toString(). Metode ini akan mengembalikan representasi String dari isi StringBuilder.

**Berikut adalah contoh penggunaan:**

*StringBuilder sb = new StringBuilder("Hello World");*

*String str = sb.toString();*

*System.out.println(str); // Output: "Hello World"*

**Mengubah StringBuilder ke dalam Class String**

Anda juga dapat mengubah StringBuilder ke dalam String dengan menggunakan constructor dari kelas String. Constructor ini menerima StringBuilder sebagai parameter dan mengembalikan objek String yang merepresentasikan isi StringBuilder tersebut.

Berikut adalah contoh penggunaan:

*StringBuilder sb = new StringBuilder("Hello World");*

*String str = new String(sb);*

*System.out.println(str); // Output: "Hello World"*

**E. Struktur Format String**

Struktur format string merupakan pola atau template string yang digunakan untuk memformat atau mengatur tampilan suatu teks atau nilai yang akan ditampilkan dalam program. Struktur ini biasanya terdiri dari karakter-karakter atau kode-kode yang berfungsi Sebagai placeholder atau tempat untuk menyisipkan nilai atau variabel pada saat program dijalankan.

Berikut ini adalah beberapa struktur format string yang umum digunakan:

%s: Merupakan placeholder untuk menampilkan string. Biasanya digunakan untuk menampilkan teks atau kata-kata.

Contoh program di java:

String name = "Alice";

System.out.printf("Hello, %s!", name);

%d: Merupakan placeholder untuk menampilkan bilangan bulat. Biasanya digunakan untuk menampilkan angka.

Contoh penggunaan di Java:

int age = 30;

System.out.printf("You are %d years old.", age);

%f: Merupakan placeholder untuk menampilkan bilangan desimal. Biasanya digunakan untuk menampilkan angka dengan koma.

Contoh penggunaan di Java:

double pi = 3.14159;

System.out.printf("The value of pi is approximately %f.", pi);

%c: Merupakan placeholder untuk menampilkan karakter. Biasanya digunakan untuk menampilkan satu huruf atau simbol.

Contoh penggunaan di Java:

char initial = 'A';

System.out.printf("The initial is %c.", initial);

%x: Merupakan placeholder untuk menampilkan bilangan bulat dalam format heksadesimal. Biasanya digunakan untuk menampilkan bilangan dalam format heksa.

Contoh dijava:

int number = 255;

System.out.printf("The number in hexadecimal is %x.", number);

Dalam Java, Anda dapat menggunakan struktur format untuk memformat output dan menyediakan argumen index untuk mengacu pada nilai variabel yang dioperasikan. Format string dengan argumen index memungkinkan Anda untuk mengontrol letak nilai variabel pada output.

- Argument Index

Berikut adalah contoh penggunaan struktur format dengan argumen index pada Java:

String name = "John";

int age = 30;

String output = String.format("My name is %1$s and I am %2$d years old.", name, age);

System.out.println(output);

Pada contoh di atas, kita menggunakan String.format() untuk memformat output. Struktur f ormat "My name is %1$s and I am %2$d years old." memiliki dua argumen index: %1$s dan %2$d.

%1$s mengacu pada argumen pertama, yaitu variabel name yang bertipe String.

%2$d mengacu pada argumen kedua, yaitu variabel age yang bertipe int.

Dalam pemanggilan String.format(), argumen pertama (name) diikuti oleh argumen kedua (age). Dengan demikian, pada output yang dihasilkan, variabel name akan muncul pada tempat %1$s, sedangkan variabel age akan muncul pada tempat %2$d.

My name is John and I am 30 years old.

- Flags: Flag adalah karakter yang digunakan untuk mengontrol tampilan format. Beberapa contoh f lag yang umum digunakan pada Java adalah:

-: Menandakan data akan ditampilkan dengan rata kiri atau left-justified.

+: Menandakan data akan ditampilkan dengan tanda plus atau minus.

0: Menandakan data akan ditampilkan dengan leading zero.

- Width: Width adalah angka yang menentukan lebar (jumlah karakter) suatu nilai atau data yang akan ditampilkan pada output. Misalnya, jika kita ingin menampilkan angka 123 pada output dengan lebar 5 karakter, maka nilai tersebut akan ditampilkan sebagai 123 (dua spasi di depan angka).

- .Precision: Precision adalah angka yang menentukan jumlah digit angka setelah tanda desimal pada nilai atau data yang akan ditampilkan pada output. Misalnya, jika kita ingin menampilkan angka 3,14159265 pada output dengan 3 digit setelah tanda desimal, maka nilai tersebut akan ditampilkan sebagai 3.142.

Contoh penggunaan struktur format pada Java:

int value = 42;

System.out.printf("Value: %-5d\n", value);

// Output: Value: 42

double pi = 3.14159265;

System.out.printf("Pi: %+.2f\n", pi);

// Output: Pi: +3.14

String name = "John";

System.out.printf("Name: %20s\n", name);

// Output- → : Name: John

double price = 9.99;

System.out.printf("Price: $%.2f\n", price);

// Output → : Price: $9.99

Dalam contoh di atas, kita menggunakan flags -, +, dan 0 untuk mengatur tampilan format width untuk menentukan lebar karakter, precision untuk menentukan jumlah digit setelah tanda desimal, dan conversion untuk menentukan tipe data atau format yang akan ditampilkan pada output.